

CROSSTALK

Nov/Dec 2015

The Journal of Defense Software Engineering

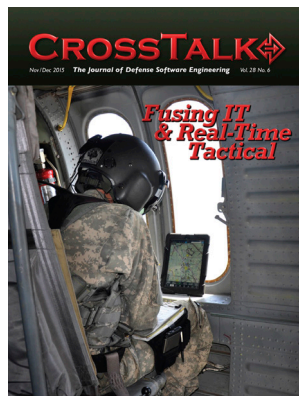
Vol. 28 No. 6

Fusing IT & Real-Time Tactical



Departments

- 3 From the Sponsor
- 38 Upcoming Events
- 39 BackTalk



Cover Design by
Kent Bingham

Fusing IT & Real-Time Tactical

- 4 Cybersecurity and Modern Tactical Systems**
A look at mitigating cybersecurity risks when interconnecting legacy embedded systems with modern-day tablets or laptops.
by C. Warren Axelrod, Ph.D.

- 12 Augmenting the Remotely Operated Automated Mortar System with Message Passing**
How the Message Passing Interface (MPI) can assist a prototype U.S. Army vehicle mounted mortar launcher system called the Automated Direct Indirect Mortar (ADIM).
by Zachary J. Ramirez, Raymond W. Blaine, and Suzanne J. Matthews

- 16 Massive Storage in a Miniature (Embedded) Package**
Examining the fundamentals of an embedded data storage system, the thoughts behind the design decisions, and different features to incorporate in an embedded data storage system.
by Anthony Massa

- 19 International Partners with Multi-Site Thin Client Interconnectivity**
With thin client and VPN technology you can save on time and travel costs while being able to more tightly integrate software and system changes.
by Brendan Conboy

- 21 Threat Modeling for Aviation Computer Security**
Threat Modeling is a technique that assists software engineers to identify and document potential security threats associated with a software product.
by Abraham O. Baquero, Andrew J. Kornecki, and Janusz Zalewski

- 28 Extending Life Cycle Models for a Repeatable Innovation Strategy**
A life cycle methodology with the necessary attributes can increase the probability for achieving a repeatable process for innovation.
by Duffy Nobles and Kevin MacG. Adams, Ph.D.

- 33 Mobile and Embedded Security Mitigations for Counterfeit Threats and Software Vulnerabilities**
Mobile and embedded software teams, users and stakeholders have historically underestimated the risk of security threats.
by Jon Hagar

CROSSTALK

NAVAIR Jeff Schwalb
DHS Joe Jarzombek
309 SMXG Karl Rogers

Publisher Justin T. Hill
Article Coordinator Heather Giacalone
Managing Director David Erickson
Technical Program Lead Thayne M. Hill
Managing Editor Brandon Ellis
Associate Editor Colin Kelly
Art Director Kevin Kiernan

Phone 801-777-9828
E-mail Crosstalk.Articles@hill.af.mil
Crosstalk Online www.crosstalkonline.org

CROSSTALK, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsor: Ogden-ALC 309 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of **CROSSTALK** providing both editorial oversight and technical review of the journal. **CROSSTALK's** mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit www.crosstalkonline.org/subscribe to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the **CROSSTALK** editorial board prior to publication. Please follow the Author Guidelines, available at www.crosstalkonline.org/submission-guidelines. **CROSSTALK** does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with **CROSSTALK**.

Trademarks and Endorsements: **CROSSTALK** is an authorized publication for members of the DoD. Contents of **CROSSTALK** are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CROSSTALK Online Services:
For questions or concerns about crosstalkonline.org web content or functionality contact the **CROSSTALK** webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CROSSTALK is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing luminpublishing.com. ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CROSSTALK would like to thank NAVAIR for sponsoring this issue.



In the beginning, there was DoD Standard 2167A for the development of software for weapon systems, and DoD standard 7935A for the development of Automated Information Systems. East was east and west was west, and never the twain would meet – or so we thought. Would we ever need the extreme rigor for requirements management, design,

documentation, configuration management, technical reviews, testing, security and safety that were used for military systems for administrative systems?

Someone thought so. In 1994, east and west not only met but became one. Military Standard 498 merged 2167A and 7935A to define a set of activities and documentation suitable for the development of both types of software development. Shortly afterward, it was cancelled as part of acquisition reform, and superseded by commercial standards such as IEEE 12207. But its legacy of fusing tactical and non-tactical systems (also known as IT – Information Technology and IS – Information Systems) remains. The world has changed much in the past 20 years, and it is clear that we do need to adjust even more to the blurring lines of what were at one times two distinctly different types of software.

The proliferation of mobile technology continues to expand through the commercial and Federal IT service marketplace. Naval Air Systems Command (NAVAIRSYSCOM) is exploring the value of transitioning “traditional” IT products into highly tactical

environments. The Electronic Kneeboard (EKB) program seeks to place cutting edge mobile technology directly in the warfighter's hands. EKB will deliver tablet technology for use in every USN and USMC aircraft in the fleet, providing real-time access to digital flight information products, imagery, and other tactical information sources. Program success is dependent upon productizing traditional Enterprise IT services, like Mobile Device Management, for bandwidth-constrained tactical scenarios.

EKB is just one of an increasing trend of modern-day tablets and laptops being used to make up for functionality and ease-of-use limitations of legacy systems. As long as modern information systems and legacy embedded systems remain independent of one another, the latter are not subject to conventional cyber attacks. However, if these systems are interconnected and interoperate, previously-avoided cybersecurity risks may be introduced. The article “Cybersecurity and Modern Tactical Systems” by Warren Axelrod in this issue looks at how these risks might be mitigated.

Another critical area that we need to manage as we fuse IT and real-time tactical systems is software safety. Military standard 882E addresses the levels of control yielded to software, and prescribes the levels of rigor that need to be applied. It is possible, and even probable, that devices that were developed for the commercial world for average consumers will also be integrated into our military systems – not just for off-line use. We need to be mindful of not only the benefits but also to apply the appropriate rigor of the new world we are in.

Al Kaniss
Software Engineering Branch Head
NAVAIR

Cybersecurity and Modern Tactical Systems

C. Warren Axelrod, Ph.D., Delta Risk LLC

Abstract. Many legacy embedded systems, such as aircraft flight-control systems and weapon fire-control systems, continue in use decades after their introduction. At the same time, we are seeing modern-day tablets and laptops being used to make up for functionality and ease-of-use limitations of legacy systems. As long as modern information systems and legacy embedded systems remain independent of one another, the latter are not subject to conventional cyber attacks. However, if these systems are interconnected and interoperate, previously-avoided cybersecurity risks may be introduced. This article looks at how these risks might be mitigated.

Background

Since the advent of digital computers more than a half-century ago, we have seen IT (information technology) and control software advance much more rapidly than underlying technologies inherent in military aircraft, ships and ground vehicles and weapons. Physical equipment may have to remain in use well beyond their anticipated decommissioning date, especially if replacements have been delayed or not approved. Such equipment will eventually contain obsolescent computer hardware and software components. Such outdated components hamper the effectiveness of their hosts. In response, programs to modernize older equipment by adding or fusing IT systems onto legacy systems are undertaken. This approach, however, introduces cybersecurity issues, which we will examine in this article.

Relative Useful Lives of Systems

Donzelli [1] describes how the operational lifespan of military aircraft has increased from about 15-20 years in the 1940s to about 40-60 years at the turn of the 21st century. The same holds true for some artillery in the author's experience. On the other hand, computer technologies generally have a much shorter lifespan, with software mostly in the 5-15 year range according to Tamai [2], and successful software lasting about 10-20 years, per Rajlich [3]. Hardware technologies and programming languages have seen new generations every decade or so since the 1940s as in Table 1.

From a categorization perspective, the first three generations, shown in Table 1, took some 30 years in total (averaging a decade for each generation of computer technology), whereas the fourth generation underlying technology (integrated circuits and microprocessors) has lasted for more than 40 years. This is somewhat misleading since there have been major advances in size reduction and lower costs for computer devices. As indicated in the column with other noteworthy events, there were also game-changing advances such as the GPS system, which

began in 1973, the World Wide Web and the introduction of Ethernet, which both began in 1980, and the adoption of mobile computing, which took off in 2005.

The underlying thesis is that in today's military aircraft, ships, vehicles, weapons and munitions can have useful lives of half-a-century or more, whereas computer equipment and software offer new generations within in a 10-20 year cycle. Consequently, one might expect the computer hardware and software to be updated between two and five times over the lifetime of the equipment. While this type of cycle is reasonable for control systems and data processing systems, it does not account for game-changing "noteworthy events" such as the Web, GPS, mobile computing and touch screens. These paradigms are often dealt with by "bolting on" additional capabilities that were not anticipated when the original systems were designed. As we shall discuss later, software engineers usually do not account for the exposure to cybersecurity attacks.

Software components and communications networks have become increasingly critical to the effective operation of mission-critical resources. Hence there is a push to transition to newer computer and communications technologies and infrastructures. However, in many cases, newer technologies have to be bolted onto legacy systems rather than being incorporated during the design, development and manufacture of software and devices.¹ The former approach results in a significantly higher cybersecurity risk, as systems, which were previously physically and logically independent, are interconnected into systems of systems [7]. It takes an understanding both of modern computer and communications technologies and the technologies incorporated in older embedded systems to be able to design and develop overall systems that demonstrate acceptable levels of safety and security [10].

Cyber-Physical Systems

A critical issue with IT systems, which did not plague embedded systems until very recently, is the high likelihood of cybersecurity compromise, which not only affects the IT systems themselves but also any other systems with which they interoperate.

To better understand what is taking place, we will examine the general structure of cyber-physical systems. NIST defines cyber-physical systems as "the tight conjoining of and coordination between computational and physical resources." Figure 1 illustrates such a relationship.

It is important to distinguish between control and administrate applications, which are usually built into embedded systems and accessed (when necessary) by administrators or operators, and data-processing or IT systems, which are separately developed or acquired applications, which are operated by internal or external end users. As long as these systems operate independently, there is little risk of cyber attack. However, it is when these systems are interfaced logically (shown by the double-ended arrow) that cybersecurity problems arise, particularly when the interoperability was not contemplated.

Legacy military real-time embedded systems, such as flight-control systems found in older aircraft and fire-control systems still operating in older weapons, were never designed to be connected to modern information systems, which are often connected to the Internet, let alone fused with them into

Table 1: Evolution of Computer-Related Resources

Generation	Period	Computer Technologies [4] [6]	Programming Languages [5] [6]	Noteworthy Events
1	1940 - 1956	<ul style="list-style-type: none"> Vacuum tubes (logic) Punched cards and printers (input/output) Magnetic drums (memory) Punched cards and paper tape (external storage) 	<ul style="list-style-type: none"> Machine language Assembly languages 	<ul style="list-style-type: none"> Computers were huge machines that contained vacuum tubes and relays that failed frequently Only very few expensive machines were available, mostly to the military and academics Data centers and their computer systems controlled by a small number of “gurus”
2	1956 – 1963	<ul style="list-style-type: none"> Transistors (logic) Magnetic tapes and disks (input/output) Magnetic core matrices (memory) Magnetic tape (external storage) 	<ul style="list-style-type: none"> High-level programming languages – FORTRAN, COBOL 	<ul style="list-style-type: none"> Commercial data centers Financial and accounting applications Increased numbers of computer programmers and systems analysts
3	1964 - 1971	<ul style="list-style-type: none"> Integrated circuits (logic and internal memory) Dumb terminals (input/output) Magnetic tapes and disks (external storage) 	<ul style="list-style-type: none"> Database languages – Sybase, Oracle User-friendly languages – BASIC, C, Pascal 	<ul style="list-style-type: none"> Remote job entry and printing Multiprocessing – two or more processors within a single computer system Multiprogramming – two or more programs on a single computer system Range of compatible machines, e.g., IBM 360 series, enabling scalability
4	1971 - Present	<ul style="list-style-type: none"> Microprocessors (logic) Keyboard & monitor (input/output) LSI and VLSI memory chips (internal memory) Magnetic tapes and disks (external storage) 	<ul style="list-style-type: none"> Object-oriented languages – Java, Ada, C++ 	<ul style="list-style-type: none"> Timesharing Virtual computing Personal computer, tablet Graphical user interface (GUI) Mouse, touch pad Grid computing Global Positioning (GPS) Internet, World Wide Web Ethernet Mobile computing
5	Present and Beyond	<ul style="list-style-type: none"> Artificial intelligence Virtual reality Quantum computers 	<ul style="list-style-type: none"> Natural languages 	<ul style="list-style-type: none"> Real-time sharing Semantic Web

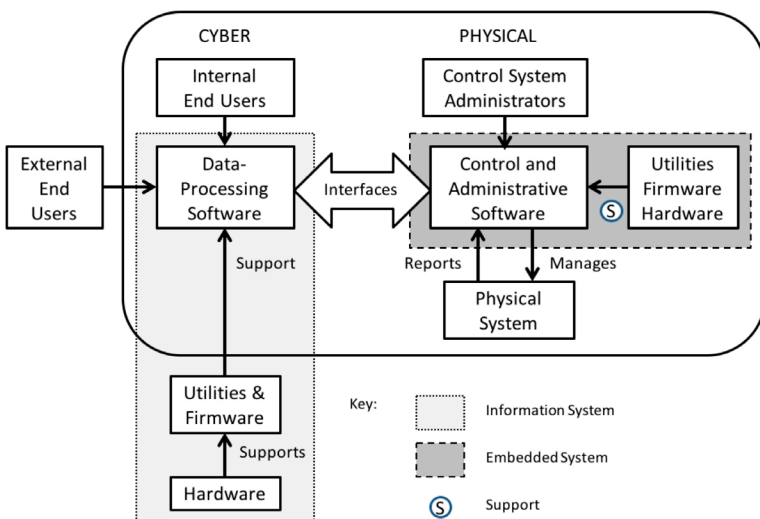


Figure 1. Cyber and Physical Components of Cyber-Physical Systems (Source: Axelrod [11])

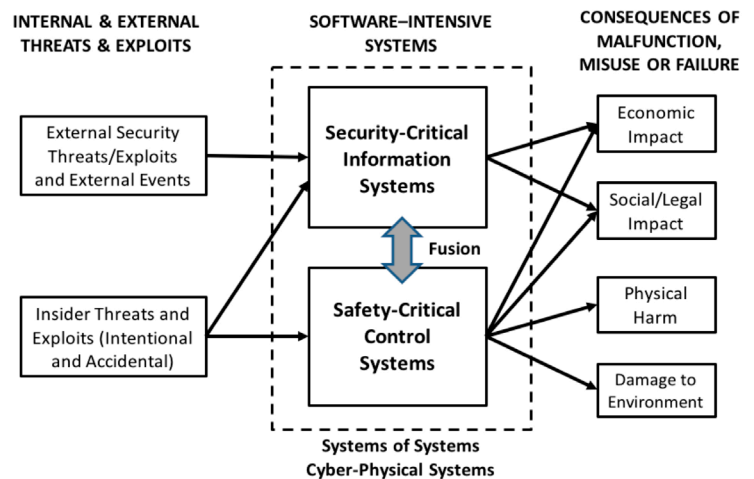


Figure 2. Threats, Exploits and Consequences for Systems of Systems (Source: Axelrod [10])

Table 2a. Threats, Vulnerabilities and Security Measures by Evolution Phase

Evolution Phase	Characteristics	Threats/Vulnerabilities	Security Measures
1. a. Mechanical b. Analog c. Electro-mechanical	<ul style="list-style-type: none"> Devices manufactured by hand by craftsmen Relatively simple designs that can be easily operated/repared Systems usually operated by mechanical and (later) electromechanical means 	<ul style="list-style-type: none"> Devices may be operated by enemy if captured Devices may behave in unexpected ways, leading to malfunction and/or failure Electromechanical components dependent on power source 	<ul style="list-style-type: none"> Place equipment in secured area Use mechanism to disable system, e.g., remove key, if system is to fall into enemy hands Train operators in security procedures
2. Digital	<ul style="list-style-type: none"> Move from analog to digital Initial hybrid systems Physically isolated systems Specialized knowledge needed to develop and operate systems Specific programming languages and software utilities Designed for particular purposes High degree of obsolescence High replacement costs Frequent failures or malfunctions Long response and repair times Use of redundancy for higher availability 	<ul style="list-style-type: none"> Insider threat –insider turned rogue Kidnapping, defection, physical and financial threats, bribery, spying and industrial espionage Inadvertent/unintentional data errors and operational mistakes Systems behave in unexpected ways, leading to malfunction and/or failure Can sometimes be used by enemy if captured 	<ul style="list-style-type: none"> Place equipment in secured area Limit electronic access via hard-wired devices placed in secured areas Monitor employees, contractors, and others (such as suppliers' staff) Remove components, e.g., hard drives from computer systems and lock in a safe in a secured area
3. Standardized	<ul style="list-style-type: none"> Standard project management Use of more generalized hardware and software platforms Isolated embedded systems Fewer programming languages, ideally one, e.g., Ada [14] 	<ul style="list-style-type: none"> Off-the-shelf software and hardware components have lower repair times since parts more readily available and more programmers trained in standard language Greater ubiquity means that more people will be familiar with products and likely know their weaknesses 	<ul style="list-style-type: none"> Place equipment in secured area or area that can be monitored Limit access to software products themselves to those who have a need, e.g., installers, system support
4. Networked	<ul style="list-style-type: none"> Linking together of proprietary systems on dedicated private networks 	<ul style="list-style-type: none"> Networked systems only as secure as their weakest link Networks open up additional threats and vulnerabilities vs. standalone systems 	<p><u>Logical</u></p> <ul style="list-style-type: none"> Use proprietary network protocols Create subnets where feasible <p><u>Physical</u></p> <ul style="list-style-type: none"> Isolate equipment, e.g., place routers in locked areas , create subnets
5. Off-the-Shelf and Open Systems	<ul style="list-style-type: none"> Use of commercially-available applications, software utilities and computer hardware Connection over private and public networks 	<ul style="list-style-type: none"> Uncertainty with respect to provenance Little or no control over supply chains Poor support of open source in some cases 	<p><u>Logical</u></p> <ul style="list-style-type: none"> Use approved and certified software Use proprietary network protocols Create subnets where feasible <p><u>Physical</u></p> <ul style="list-style-type: none"> Use approved and certified hardware Isolate equipment, e.g., place routers in locked areas

tightly-bound cyber-physical systems. Yet, as we have discussed, there is an increasing need to extend the useful lives of legacy resources that continue in service well beyond their expected useful lives.

There are a number of ways in which the useful life of legacy software systems can be extended in order to avoid having to replace existing systems or bolt on modern front-end systems.² However, the use of these latter systems, many running on personal computers and tablets that are in turn connected to the Internet, is inevitable since organizations, such as the DoD, cannot afford to provide required functionality via the traditional approach of building the systems in-house.

Security and Safety of Cyber-Physical Systems

When IT systems (which are connected to private and public networks) and embedded systems (which historically have been standalone with restricted access) are interfaced, one with the other, the overall system of systems is vulnerable to threats typical of both types of system and subject to the consequences of both hacking of front-end IT systems and the malfunctioning and failure of the back-end control systems. This situation is illustrated in Figure 2.

The diagram shows that security-critical information systems, which are connected to public networks (such as the Internet), are affected by both external and internal threats and exploits,

Table 2b. (continued from 2a) Threats, Vulnerabilities and Security Measures by Evolution Phase

Evolution Phase	Characteristics	Threats/Vulnerabilities	Security Measures
6. Web Applications	<ul style="list-style-type: none"> Search Email, messaging Posting of blogs/comments E-commerce 	<ul style="list-style-type: none"> Significant exposure to others on the Internet Vulnerable to social engineering, e.g., phishing, links to dangerous websites 	<p><u>Logical</u></p> <ul style="list-style-type: none"> Implement threat modeling Subscribe to exploit, vulnerability, and incident information sharing and analysis services Invoke static and dynamic testing Use white listing and black listing of websites Use deception (see [15] – honey pots, fake systems) <p><u>Physical</u></p> <ul style="list-style-type: none"> Install security software, e.g., antivirus, firewalls Use strong authentication, e.g., biometrics Enable tracking and “wipe” software Turn off equipment when not in use, where feasible Only use approved computer hardware and software
7. Mobile	<ul style="list-style-type: none"> CMDs (Commercial Mobile Devices) typically use broadband or Wi-Fi to connect to the Internet 	<ul style="list-style-type: none"> Significant exposure to others on the Internet Vulnerable to social engineering, e.g., phishing, links to dangerous websites 	<p><u>Logical</u></p> <ul style="list-style-type: none"> Threat modeling Exploit, vulnerability, and incident information sharing and analysis Static and dynamic testing Only use approved apps <p><u>Physical</u></p> <ul style="list-style-type: none"> Install security software, e.g., antivirus, firewalls Use strong authentication, e.g., biometrics Enable tracking and “wipe” software Turn off equipment when not in use, where feasible Only use approved equipment
8. Cloud Computing	<ul style="list-style-type: none"> Software-as-a-Service (SaaS) Platform-as-a-Service (PaaS) Infrastructure-as-a-Service (IaaS) 	<ul style="list-style-type: none"> Exposure generally to others on the Internet Vulnerable to social engineering Exposure to others using same cloud service provider Legal disclosure Interoperability issues Portability (difficulty in changing services) Scalability 	<p><u>Logical</u></p> <ul style="list-style-type: none"> Ensure effective partitioning, i.e., isolation from other customers Implement security measures Ensure privacy is protected <p><u>Physical</u></p> <ul style="list-style-type: none"> Isolate equipment, e.g., place routers in locked areas

whereas safety-critical control systems traditionally were minimally affected by external threats, if at all. Also, designers and developers of security-critical information systems formerly were not concerned about their systems causing physical harm or damage to the environment. However, when information and control systems are interconnected, they inherit both the positive and negative characteristics of both. In particular, information systems might be a conduit for malware into control systems and information systems take on some of the liability for malfunctions or failures of the control systems. It appears that the bulk of the responsibility for protecting the overall

software environment lies with the information systems since they present the pathways for malicious activities. Nevertheless, software engineers need to understand the implications of adverse behavior of the control systems since they must focus their attention on protecting against particular events, such as crashes of vehicles or inaccurate aim of weapons.

Brief History of Real-Time Tactical Computer Systems

The evolution of real-time naval tactical digital computer systems began with the transition from analog systems in the early

1960s with the Naval Tactical Data System (NTDS) according to David Boslaugh's detailed accounts [12], [13]. The original digital computers were standalone minicomputers connected to analog servomechanism-based control systems.

In Tables 2a and 2b we show the characteristics of this and subsequent phases of the evolution of such systems, the threats and vulnerabilities that make up the risks to and from them and what measures can be put in place to mitigate those risks.

Beginning with Phase 5, we see that systems become exposed to increasing outside threats due to connection to public networks such as the Internet, initially through separate systems, but increasingly with interconnected and interoperating architectures.

Certification

Certification is mandatory for software aboard commercial aircraft, for example. The basic certification standard used is DO-178C [16], which superseded DO-178B in January 2012. This standard has been adopted by the DoD as guidance for certifying military avionics [17].

The DO-178C certification standard categorizes types of software as to the severity of the consequences if the system were to fail. This is shown in Table 3.

This clearly shows that the standards are much more stringent for flight control and management systems, as would be expected since the consequences of failure are usually catastrophic. Onboard information systems, on the other hand, are shown to have minimal consequences. As long as the information systems, which include tablets used by pilots for navigation purposes, which have caused the grounding of commercial aircraft [18], are kept separate from the control systems, such classification appears to be reasonable. However, as soon as links between the two are created, then a cyberattack can have catastrophic consequences. The risk from on-board entertainment systems may be less for military aircraft compared to civilian commercial planes, but the risk from CMDs is likely to be

the same if not greater for military aircraft.

In Figure 3 we illustrate how cyber-physical systems can be shown as layers with IT systems at the perimeter and control systems at the center. The various levels can be entered by certain authorized groups and by attackers if the systems as fused together since there may not be effective barriers to entry or for exfiltrating sensitive information and control data.

It is important to consider such systems of systems holistically from both the security and safety perspectives which can be achieved only if information security professionals and safety engineers work collaboratively throughout the system development lifecycle, including operation, updating and decommissioning.

Cybersecurity Risk of Safety-Critical Systems

The DoD chief information officer, Teri Takai, announced on March 12, 2014 that DIACAP (DoD Information Assurance Certification and Accreditation Process) was to be replaced as of that date by the NIST (National Institute of Standards and Technology) risk management framework governed by NIST Special Publications SP 800-37, SP 800-39 and SP 800-53 [19].

NIST SP 800-53 [20] provides a three-tiered risk management approach that addresses strategic and tactical risk at the corresponding organization, mission/business process and information system levels.

A Risk Management Framework (RMF) is presented in SP 800-53. The RMF consists of six steps as follows:

- Categorize information systems based on impact assessment
- Select the applicable security control baseline
- Implement the security controls and document their design, development and implementation details
- Assess security controls as to their meeting security requirements
- Authorize information system operation
- Monitor security controls



CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, **CROSSTALK** can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for the areas of emphasis we are looking for:

Integration and Interoperability

May/Jun 2016 Issue

Submission Deadline: Dec 10, 2015

CMMI - The Agile Way

Jul/Aug 2016 Issue

Submission Deadline: Feb 10, 2016

Supply Chain Risks in Critical Infrastructure

Sep/Oct 2016 Issue

Submission Deadline: Apr 10, 2016

Please follow the Author Guidelines for **CROSSTALK**, available on the Internet at <www.crosstalkonline.org/submission-guidelines>. We accept article submissions on software-related topics at any time, along with Letters to the Editor and BackTalk. To see a list of themes for upcoming issues or to learn more about the types of articles we're looking for visit <www.crosstalkonline.org/theme-calendar>.

Table 3. RTCA/DO-178C standard applied to aircraft certification

System	Type of System	Level A (Catastrophic)	Level B (Hazardous)	Level C (Major)	Level D (Minor)
Flight control	Control	X			
Cockpit display and controls	Control	X			
Flight management	Control	X			
Brakes and ground guidance	Control		X		
Centralized alarms management	Information			X	
Cabin management	Information				X
Onboard communications	Information				X

As described in [10] for software systems generally and in [21] for avionics software, security requirements have to be inserted early in the software development lifecycle and carried through design, development, testing and implementation.

In [21] the author quotes Robert Dewar, president of Ad-aCore, as saying “We have fortunately not experienced an aircraft accident where a software bug has resulted in loss of human life.” Sadly, there was a recent accident in which an Airbus A400M military cargo and troop transport plane crashed on a test flight on May 9, 2015 and resulted in the deaths of four persons. Several weeks later it was revealed that the crash was caused by faulty software installation [22].³

Kidnapping, Defection, Threats, Bribery, Spying and Industrial Espionage

One might ask why these topics are mentioned in an article about cybersecurity. Surely today's major concerns relate to cyber rather than physical attacks? Don't such methods as kidnapping belong to a bygone era when computers operated in isolation in guarded data centers and there was no way for outsiders to access the systems? The reality is that there may often be a physical component even when the most visible aspect of an attack is via the Internet.

As far back as the 1960s and 1970s even civilian computer security experts were concerned about being kidnapped by foreign powers to gain access to their specialized knowledge of “system internals,” the underlying software that controls the operation of computers.⁴ Since the early 1990s there have been a number of movies, such as *Sneakers*, (1992), *Swordfish* (2001), *Firewall* (2006) and *Live Free or Die Hard* (2007), that were based on the idea that experts could be kidnapped by criminals and forced to give up information on how computer systems in the government and private sectors could be used for nefarious purposes.

While reporters are quick to publicize major cyber attacks against computer systems, where huge amounts of sensitive personal data and intellectual property are obtained, as being the work of hackers in Russia, North Korea, Iran, China and the like, few mention that many attacks are facilitated by insider knowledge of the systems and analysis of the stolen data requires subject-matter expertise. This expert knowledge can only be obtained from insiders, former employees or contractors, whether voluntarily (by defectors), involuntarily (from kidnapping, threats) or accidentally (via social engineering).

Therefore, when it comes to protecting modern tactical systems, one must not only consider the possibility of hacking into operational systems and taking over control systems, but also consider cyber and physical attacks against defense and

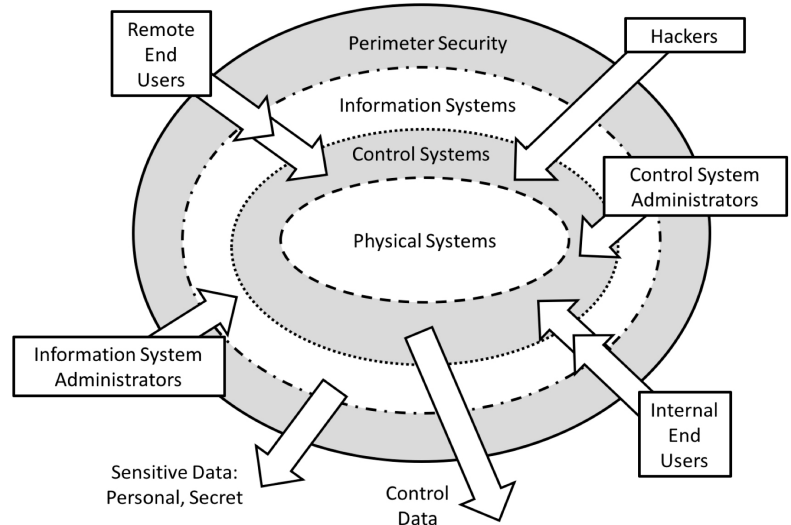


Figure 3: The Systems “Onion” Showing Access and Data Exfiltration

intelligence agency and contractor systems and personnel, and against former employees and contractors, to obtain information about the design, programming and operation of tactical systems.

Conclusion

Perhaps the best way to ensure that real-time tactical systems are not subjected to cyber attacks is to keep them separate from IT systems, particularly those IT systems that use commercial and open-source software and hardware. However, this is increasingly less feasible for technological and economic reasons. We just have to face the fact that there is pressure to use relatively inexpensive off-the-shelf technology and free open-source software components and to interface these systems, via loose coupling or tight interoperability, with legacy systems in order to attain desired levels of functionality and usability.

Given this situation, it is important to take a proactive stance by examining the cybersecurity impact of each and every change to cyber-physical system environments, rather than just succumb to pressure and respond to problems as they occur. In addition, if modern IT systems, particularly those that access the Internet or utilize cloud-computing services, are to be integrated with legacy real-time tactical systems so that they interoperate, then it is necessary to go through extensive validation and verification of the combined system to ensure that the tactical control systems cannot be compromised by someone entering the combined system via the front-end IT system.

ABOUT THE AUTHOR



Dr. C. Warren Axelrod is a senior consultant with Delta Risk LLC, a Chertoff Group Company, specializing in cyber security, risk management and business resiliency. Previously, he was the Business Information Security Officer and Chief Privacy Officer for US Trust. He was a founding member of the FS/ISAC (Financial Services Information Sharing and Analysis Center) and represented national financial services cyber security interests during the Y2K date rollover. He testified before Congress in 2001 on cyber security. His recent books include Engineering Safe and Secure Software Systems (Artech House, 2012) and Outsourcing Information Security (Artech House, 2004). He holds a Ph.D. in managerial economics from Cornell University, and a B.Sc. in electrical engineering and an M.A. in economics and statistics from Glasgow University. He is certified as a CISSP and CISM.

Phone: 917-670-1720

E-mail: waxelrod@delta-risk.net

NOTES

1. In order to extend the useful lives of various munitions, GPS and inertial guidance systems and adjustable tail fins were added to existing projectiles and bombs. These add-on features provide much greater precision in hitting a target and may also extend the range. Two examples are the Excalibur 155 mm precision-guided artillery shell [8] and the JDAM bomb [9]. In these cases, existing "dumb" munitions are retrofitted with the requisite technologies.
2. A number of these methods, such as using software wrappers and executing legacy code on modern microprocessors, are described in an article in the December 2001 issue of CrossTalk, which has the title "Software Legacy Systems: Bridging the Past to the Future." This issue is available at <<http://www.crosstalkonline.org/back-issues/>>
3. In August 2005, the author happened to be on a brand-new cruise ship leaving St. Petersburg. The ship suddenly stopped and did not move again for about five hours. The captain, in an attempt to assuage passenger concerns, reported that we had not run aground but that the engines had failed due to a "software problem." As more vehicles are built with software-managed and "fly-by-wire" control systems, this type of problem will surely become much more common. While the risk to cruise-ship passengers may be small, the same claim cannot be made for aircraft, trains or road vehicles.
4. The author was aware, in the 1970s, of a group of a dozen experts in a particular computer system who were so concerned about being kidnapped that they organized a formal contact system. In one case, a member of the group was at an airport, about to board a plane, when he called his colleagues. The latter validated his concerns about whom he was supposedly meeting and he cancelled the trip.

Hiring Expertise



Engineers and Computer Scientists

The Software Maintenance Group at Hill Air Force Base is recruiting **civilians** (U.S. Citizenship Required). Benefits include paid vacation, health care plans, matching retirement fund, tuition assistance, paid time for fitness activities, and workforce stability with 150 positions added each year over the last 5 years.

Become part of the best and brightest!

Hill Air Force Base is located close to the Wasatch and Uinta mountains with skiing, hiking, biking, boating, golfing, and many other recreational activities just a few minutes away.



Send resumes to:

309SMXG.Recruiting@us.af.mil

or call (801) 777-9828



www.facebook.com/309SoftwareMaintenanceGroup

REFERENCES

1. Donzelli, Paolo, and Roberto Marozza. "Customizing the Software Process to Support Avionics Systems Enhancements," *CrossTalk*, 4.9, (September 2001):10-14.
2. Tamai, Tetsuo, and Yohsuki Torimitsu, Software Lifetime and its Evolution Process over Generations. Proc. of the Conf. on Software Maintenance. Durham, UK, 1992: 63-69. < <http://tamai-lab.ws.hosei.ac.jp/pub/icsm92.pdf>> 7 June 2015.
3. Rajlich, Václav "Software Lifespan Models," Chapter 2 in Václav Rajlich (ed.) *Software Engineering: The Current Practice*. Boca Raton, FL: Chapman and Hall/ CRC, 2011. 19-30.
4. Beal, Vanie. "The Five Generations of Computers," *Webopedia*, 22 April 2015. <http://www.webopedia.com/DidYouKnow/Hardware_Software/FiveGenerations.asp>. 7 June 2015.
5. "The Five Generations of Software," *RBV Web Solutions*, 20 March 2000. <<http://www.rbvweb.net/software.html>> 7 June 2015.
6. "The Four Generations of Computer Hardware," *RBV Web Solutions*, 10 December 2005. <<http://www.rbvweb.net/generations.html>> 7 June 2015.
7. Baldwin, Kristen, Judith Dahmann, and Jonathan Goodnight, "Systems of Systems and Security: A Defense Perspective," 14.2 INCOSE Insight (July 2011): 22-25. <<http://www.acq.osd.mil/se/docs/SoS-Security-INCOSE-INSIGHT-vol14-issue2.pdf>>
8. "M982 Excalibur," *Wikipedia*. <http://en.wikipedia.org/wiki/M982_Excalibur> 7 June 2015
9. Harris, Tom. "How Smart Bombs Work" 20 March 2003. *Howstuffworks.com*. <<http://science.howstuffworks.com/smart-bomb.htm>> 7 June 2015
10. Axelrod, C. Warren. *Engineering Safe and Secure Software Systems*, Norwood, MA: Artech House, 2012.
11. Axelrod, C. Warren. *Mitigating the Risks of Cyber-Physical Systems*, Proc. of the 2013 IEEE LISAT Conf., Farmingdale, NY, 2013.
12. D.L. Boslaugh, "First-Hand: No Damned Computer is Going to Tell Me What to DO - The Story of the Naval Tactical Data System, NTDS. *Engineering and Technology History Wiki* <http://ethw.org/First-Hand:No_Damned_Computer_is_Going_to_Tell_Me_What_to_DO_-_The_Story_of_the_Naval_Tactical_Data_System,_NTDS> 7 June 2015
13. D.L. Boslaugh, *When Computers Went to Sea: The Digitization of the United States Navy*, Los Alamos, CA: IEEE Computer Society, 1999.
14. Brosgol, Benjamin M. "Ada 2005: A Language for High-Integrity Applications." 19.8 *CrossTalk*, (August 2006): 8-11.
15. Almeshekeh, Mohammed H., and Eugene H. Spafford, "Using Deceptive Information in Computer Security Defenses." 4.3 *Int. J. of Cyber Warfare and Terrorism*, (July-September 2014): 63-80.
16. "DO-178C," *Wikipedia*, 5 January 2012. <<http://en.wikipedia.org/wiki/DO-178C>> 7 June 2015.
17. Romanski, George. "The Challenges of Software Certification," *CrossTalk* 14.9, (September 2001): 15-18.
18. BBC News, "American Airlines planes grounded by iPad app error," 29 April, 2015. <<http://www.bbc.com/news/technology-32513066>> 7 June 2015
19. Perera, David. "DoD Abandons DIACAP in Favor of the NIST Risk Management Framework." *FierceGovernmentIT*. < <http://www.fierceregovernmentit.com/story/dod-abandons-diacap-favor-nist-risk-management-framework/2014-03-18>> 7 June 2015
20. National Institute for Science and Technology. *Security and Privacy Controls for Federal Information Systems and Organizations*, NIST Special Publication 800-53 Revision 4, April 2013 (updated as of 22 January 2015). < <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf>> 7 June 2015
21. Howard, Courtney E. "Safety- and security-critical avionics software." *Military & Aerospace Electronics*, 1 February, 2011. <<http://www.militaryaerospace.com/articles/print/volume-22/issue-2/technology-focus/safety-and-security-critical-avionics-software.html>> 7 June 2015
22. Wheatley, Mike. "Faulty Software Install Led to Airbus A400M Plane Crash," *SiliconAngle*, 1 June 2015. < <http://siliconangle.com/blog/2015/06/01/faulty-software-install-led-to-airbus-a400m-plane-crash/>> 7 June 2015.

FURTHER READING

1. Axelrod, C. Warren. "Trading Security and Safety Risks within Systems of Systems," 14.2 INCOSE Insight (July 2011): 26-29.
2. Corman, David. "The IULS Approach to Software Wrapper Technology for Upgrading Legacy Systems, 14.12 *CrossTalk*, (December 2001): 9-13.
3. DoD Inspector General, *Improvements Needed with Tracking and Configuring Army Commercial Mobile Devices*. Report No. DODIG-2013-060, March 26, 2013.
4. Hecht, Herbert. *Systems Reliability and Failure Prevention*, Norwood, MA: Artech House, 2004.
5. Joyce, Robert R. *History of the AN/UYK-20(V) Data Processing System Acquisition and its Impact on Tactical Systems Development*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1976.
6. Kölle, Raine, Garick Markarian, and Alex Tartar, *Aviation Security Engineering: A Holistic Approach*, Norwood, MA: Artech House, 2011.
7. Kornecki, Andrew J., and Janusz Zalewski. "The Qualification of Software Development Tools from the DO-178B Certification Perspective," 19.4 *CrossTalk*, (April 2006): 19-22.
8. Kornecki, Andrew J., and Janusz Zalewski. "Certification of Software for Real-Time Safety-Critical Systems: State of the Art." 5.2 *Innovations in Systems and Software Engineering*, (June 2009): 149-161.
9. Leveson, Nancy G. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press, 2011
10. Littlejohn, Kenneth, Michael V. DelPrincipe, Jonathan D. Preston, and Ben A. Calloni. "Reengineering: An Affordable Approach for Embedded Software Upgrade," 14.12 *CrossTalk*, (December 2001): 4-8.
11. Luke, Jahn A., Douglas J. Halderman, and William J. Cannon. "A COTS-Based Replacement Strategy for Aging Avionics Computers, 14.12 *CrossTalk* (December 2001):14-17.
12. Rosengard, Phillip I. *Avionic Computer Software Interpreter*, Patent No. US6564241, May 13, 2003.
13. Spitzer, Cary R. Ed. *Avionics: Elements, Software and Functions*, Boca Raton, FL: CRC Press, 2007.
14. Swassing, Margaret M. "A Brief History of Military Avionics." 2014 SFTE (The Society of Flight Test Engineers)/SETP (The Society of Experimental Test Pilots) Southwest Symposium, Fort Worth, TX, October 24-26, 2014.

Augmenting the Remotely Operated Automated Mortar System with Message Passing

Zachary J. Ramirez, United States Military Academy
Raymond W. Blaine, United States Military Academy
Suzanne J. Matthews, United States Military Academy

Abstract. This paper looks at how the Message Passing Interface (MPI) can assist a prototype U.S. Army vehicle mounted mortar launcher system called the Automated Direct Indirect Mortar (ADIM). The ADIM's capabilities are augmented by the Remotely Automated Mortar System (ROAMS) by enabling fuzees to be set remotely. The performance of the initial ROAMS prototype, a threaded Python server using Raspberry Pis, is limited by Python's Global Interpreter Lock (GIL). In this paper, the prototype is redesigned using MPI and the C programming language to dramatically improve the efficiency of the system.

1. Introduction

The U.S. Army is developing a system called the Automated Direct Indirect Mortar (ADIM) system [1]. The ADIM is mounted to a high mobility multipurpose wheeled vehicle (HMMWV) and fires belt-fed 81mm mortar rounds. This system increases the capability of the conventional mortar by adding some key features. The most important features are the speed at which it can fire, stabilize, and re-fire, and the ability to conduct "shoot and scoot" missions. Shoot and scoot missions provide a key advantage, allowing the mortar operators to fire and leave the area before an enemy can acquire their location via radar and counter fire. This increased capability of lethality, provided by the rate of fire, survivability, and increased mobility, is essential to maintaining our technological superiority on the battlefield. However, the ADIM cannot be used to its full potential because of a limitation with current 81mm mortar rounds. Current mortar rounds must have their fuzees manually set prior to being loaded into the ADIM. This requires the system to be unloaded if the desired fuze setting is not available in the magazine, severely limiting the speed of operation.

In order for the ADIM system to reach its full and future potential, 81mm rounds must have several key qualities that current munitions lack. First, they must be "smart", accepting GPS locations allowing for flight alteration and precision fires. Second, the round's fuze setting must be able to be set and changed remotely. These capabilities do not currently exist. Additionally, the system must have a user-friendly interface, the rounds must be initially powered from the battery of a HMMWV, and they must retain power for the duration of flight.

An undergraduate capstone project at the United States Military Academy (USMA) called Remotely Operated Automated Mortar System (ROAMS) attempted to tackle these shortcomings during the 2013-2014 academic year. In the first iteration of this multi-year project, the focus was to optimize the fuze setting remotely.

The ROAMS system uses Raspberry Pis to simulate the hardware of the magazine server and individual smart-round mortars. The Raspberry Pi [2] is a popular, credit-card sized single-board computer (SBC) that retails for \$35.00. The choice of Raspberry Pi enables the design team to cheaply prototype the hardware that would eventually be included in a custom integrated circuit (IC) for the smart rounds. Due to its ease of programmability, Python was selected as the language of choice to program the magazine server. The magazine server communicated with the smart-round Raspberry Pis using Python threads, simulating a classic "client-server" system.

The initial prototype worked well when interfaced with a test system consisting of four mortar rounds. However, the program's responsiveness and usability decreased as the number of fuze clients increased. This is especially problematic as the ADIM mortar chamber has a 20-round capacity. The responsiveness issue is particularly troubling because on the battlefield, every second counts. This work attempts to tackle this problem by redesigning the ROAMS system to support efficient remote fuze-setting.

This paper analyzes a redesign of the ROAMS magServer-client system using the Message Passing Interface (MPI) [3] and the C language. In order to test the scalability of the ROAMS system, a cluster of 21 Raspberry Pis was built to simulate the full ADIM system. We measure the performance of the MPI system and compare it to the client-server system implemented in Python. The MPI implementation results in a time reduction of up to 90 percent of the original Python prototype, suggesting that MPI is a promising technique to improve the speed of remote fuze setting.

The rest of the paper is organized as follows. Section II gives a detailed overview of the original ROAMS prototype, its key limitation, and motivations to transition to C and MPI. Section III describes the redesign of ROAMS to use MPI. Finally, preliminary results and conclusions are presented in Sections IV and V respectively.

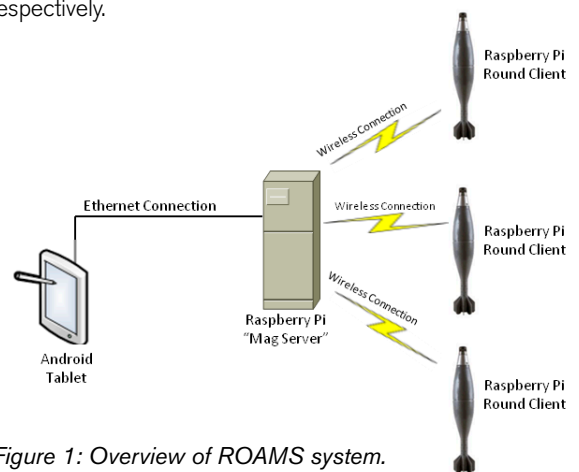


Figure 1: Overview of ROAMS system.

2. Overview of the ROAMS System

Figure 1 illustrates a simplified network layout of the ROAMS system prototype. A Raspberry Pi is used to simulate the microprocessor needed in each smart-round, and a central Raspberry Pi acts as the magazine server (or magServer). An Android tablet serves as an interface for soldiers to control the magServer and, by extension, the

mortar rounds themselves. The magServer as originally designed has four jobs:

- Establish connection with every mortar and store its state information.
- Establish connection with the tablet and provide an On-Demand list of mortars at its disposal.
- Accept and relay all commands from the user to the rounds.
- Provide setting verification on all rounds in its control.

The Android tablet accomplishes this by displaying the list of available mortar rounds for the user to select, and allows them to change the fuze setting and or GPS data. The magServer begins by setting up both a wired and wireless network interface. The server then connects to the user's Android interface device on the wired interface, and listens for mortar round connections on the wireless interface. When a mortar round is connected, the magServer adds it to the inventory and transmits to the tablet an updated list of mortar rounds. The server maintains a list of all rounds in its magazine, as well as their specific attributes (such as fuze setting and GPS coordinates).

The ROAMS remote fuze communication system was set up using a series of sockets following a classic server-client relationship. The magServer acts as the focal point between the client fuzes and the user interface. Whenever the server starts up, it runs a single-threaded Python script that accepts connections from the fuzes and the user interface. It then maintains a list of fuzes—with their relevant information—dynamically and sends update information out from the user interface. It also keeps the user interface updated on any change in fuze settings.

Python Limitations

A key limitation of the original ROAMS system was its use of Python to implement client-server threading. While a very popular language, Python is a very inefficient choice for multithreaded applications. This was highlighted in the late 2000's by David Beazley, who implicated Python's Global Interpreter Lock (GIL) as the source of its performance issues [4]. The GIL essentially forces Python programs to only run one thread at a time, even if a Python program is multi-threaded. This design decision exists to enforce memory safety in the Python interpreter.

Consequently, a program running two Python threads can run twice as slow as a Python program running a single thread. The Python community has resisted calls to remove the GIL, as doing so will reduce the safety of Python applications and reduce the speed of single threaded programs. All of these reasons suggest that Python is (for the immediate) a poor choice for creating a multi-threaded application.

Transition to C and MPI

These limitations forced the design team to explore other languages to better support multi-threading. The team settled on the C language, mainly due its native support for multi-threading, which is executed at the operating system level. While the onus for enforcing memory and thread-safety rests solely on the shoulders of the developer, C allows for more opportunities to enhance performance.

While C fully supports network socket programming over TCP/IP, the Message Passing Interface (MPI) library is used to enable the magServer to communicate with the individual clients. MPI is a standard in the high performance computing world, and is designed to enable efficient and scalable communication between multiple computers. The MPI library also has support for asynchronous communication and collective communication operations, which can drastically increase the rate at which messages are sent and received.

3. Methods

Figure 2 shows the custom 21-node Raspberry Pi B+ cluster built to simulate the full ADIM system. Each node in this cluster requires a USB wireless adaptor to both broadcast and receive wireless signals, similar to the intended implementation. Each node uses a 4GB microSD card to run the Linux operating system and store magServer and smart-round client program files. The cluster also requires a power supply to replicate the HMMWV battery for each node.



Figure 2: Final Raspberry Pi cluster.

A custom power supply was built for the project that provides surge protection, voltage conversion, and eliminates the need for 21 separate power cords. The custom case design enables the entire system to be passively cooled. The magServer node also requires a special wireless adapter to host the wireless network. Cluster and implementation details are discussed in detail below.

Cluster Configuration Details

The master Raspberry Pi node acts as a wireless access point (WAP) and dynamic host configuration protocol (DHCP) server for the project using instructions procured from the Raspberry Pi HQ website [5]. The South Hampton Raspberry Pi cluster tutorial [6] was a starting point to set up MPI on our cluster.

This application uses a custom DHCP server to assign IP addresses to each node in the cluster, requiring some additional configurations not outlined in the South Hampton tutorial. For example, the SSH configuration file was modified to disable reverse DNS lookup. Next, a Python script was added to send each worker's IP address to the magServer when the system initially boots up. This enables the magServer to automatically know at start-up the number of available worker nodes (active rounds) and their respective IP addresses.

ROAMS MPI Implementation

In the context of ROAMS, the magServer can be thought of a “master” node that passes messages to a series of “worker” nodes (smart rounds) in the ADIM magazine. Upon start up, the magServer has a list of the available “active” rounds in the magazine. Each message sent from the magServer to a particular smart round contains a set of commands to set its fuze. Each worker, upon receiving its message and setting its fuze, sends back a confirmation message.

For the scope of this paper, the design uses point-to-point communicators MPI_Send and MPI_Recv to implement the communication model. The MPI_Send function enables the magServer to send a message to a worker node. The MPI_Recv function allows a worker node to receive a message from the magServer. Thus, a pair of MPI_Send/MPI_Recv communicators is necessary each time a message is sent from the magServer to the worker nodes, or vice versa.

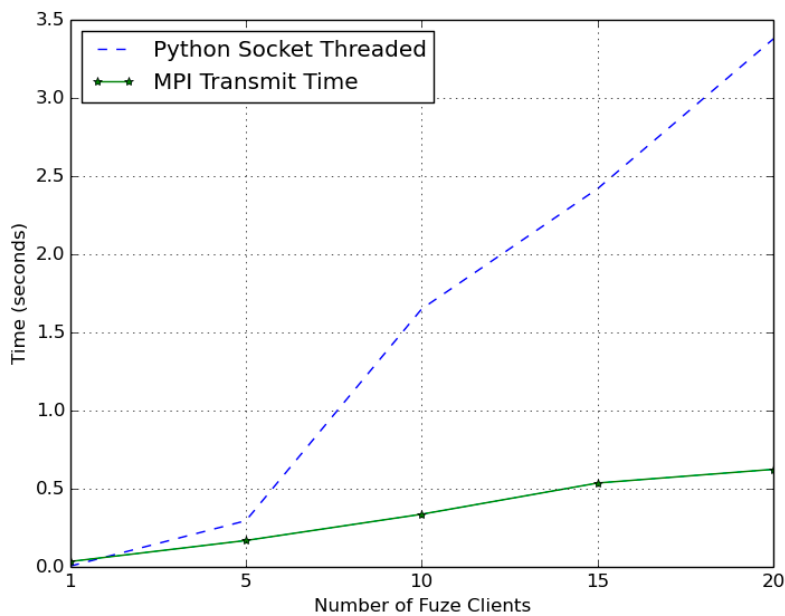


Figure 3: Time spent sending orders to new fuzes.

4. Results

The scalability of MPI compared to the Python client-server program is benchmarked by measuring two stages of execution: the time taken to send a message indicating a change in one or more clients' status (Figure 3), and the time taken to receive acknowledgement from the fuze clients that the change was made and implemented (Figure 4).

These experiments do not consider the time needed to communicate to the user interface, as scaling efficiency issues are not applicable in this context. The experiments also don't reflect the amount of time needed to acquire fuze clients during operation. This is due to the current system's inability to properly simulate when a mortar is fired. The conclusion section includes a discussion detailing what a proper future simulation of the process will look like, and some hypotheses on running time.

For each execution stage, the running time of the threaded Python implementation is compared against the MPI version. To illustrate scalability, the run time is measured as the number of

supported fuze clients is increased from one to twenty, in increments of five. We measure the percentage of run-time reduction by use the equation $(1-M/P) \times 100$ where M and P are the execution times of the MPI and Python implementations, respectively.

Sending a Message to Fuze Clients

Figure 3 shows the average time it takes each implementation to send fuze data to all the clients. In this particular execution stage, the Python implementation performs moderately well, with execution time ranging from 0.00267 seconds on a single fuze to 3.37702 seconds on twenty fuzes. While the MPI implementation also experiences a modest increase in running time, it takes 0.03241 on a single fuze and 0.62245 seconds on twenty, requiring less than a second to compute regardless of the number of fuzes. This represents an 81.56 percent reduction in time for transmitting messages to the full twenty rounds.

Receiving Confirmation from Fuze Clients

Figure 4 depicts the average time it takes the Python version to receive confirmation from all the fuze clients compared to the MPI implantation. When dealing with five fuzes, it takes 1.529 seconds for the Python implementation to receive confirmation. However, as the number of fuzes increases to fifteen, the Python threaded version takes on average 5.677 seconds. At twenty clients, it takes the Python implementation 8.337 seconds on average. In contrast, the MPI implementation takes 0.05375 seconds on average to receive confirmation from a single fuze, 0.74295 seconds for fifteen fuzes, and 0.83812 seconds for twenty. This corresponds to reduction in running time of 89.95 percent.

5. Conclusion

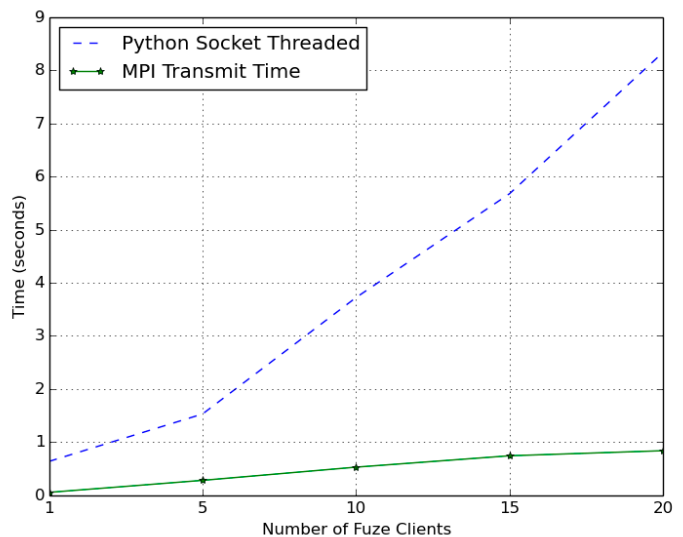


Figure 4: Time required to receive confirmation from fuzes.

The experimental results clearly show the benefit of using the MPI implementation for remotely setting fuze clients on ROAMS. Using MPI allows ROAMS to reduce the time necessary to acquire fuze information by up to 89.95 percent, corresponding to a speed up of 10.54. In all execution stages, it takes MPI less than a second to perform the desired task, regardless of the number of fuzes. In contrast, the Python implementation can take up to ten seconds.

While the difference may seem marginal on the surface, every

ABOUT THE AUTHORS

second counts on the battlefield. The current ADIM system has a fire rate of 30 rounds per minute, or a round every two seconds. Therefore, the reductions from 3.3 and 8.3 seconds to less than a second correspond to possibly two to four more rounds directed at the enemy. When a soldier is in contact with the enemy in a firefight, two to four mortar rounds could be the difference between achieving the objective and failing to suppress the enemy. In order for the ADIM to be useful, a soldier needs to know as soon as possible that the changes were received and his equipment is ready for use so he can continue to react to the ever changing battlefield.

These results also indicate the superiority of using MPI to achieve system scalability. MPI is a long-standing standard for a reason. The experimental results clearly show that it is faster than standard Python sockets for broadcasting and receiving messages from many nodes. We encourage other developers designing server-client systems to explore MPI as a potential library to improve performance. Future work will explore other MPI operations, such as collective and asynchronous communication constructs, in a further effort to improve performance in the ROAMs system.

Notably, the current experimentation does not include the amount of time needed to maintain the list of active fuze clients. As each round is fired from the chamber, it becomes "inactive." When a new mortar is inserted into the chamber, the round becomes "active." In both cases, the magServer needs to know about the change of status in individual rounds to maintain an accurate list of the mortars available at any given time.

Message passing can assist in keeping the magServer updated as follows. Every time a mortar is fired, it sends a message to the magServer indicating that it is no longer active. The magServer, upon receiving the message, will need to remove the mortar's IP address from the list of "available" IPs. When a new mortar is added to the chamber, it sends a message to the magServer notifying that the round is active. Upon receiving the message, the magServer adds the new IP address to the list of "available" IPs. Regardless of whether a new round is "acquired" or "disabled/fired", the cost is a single send/receive operation plus the time needed to update the list.

The preliminary results suggest that the time needed to send/receive a single message using MPI takes between 0.03 and 0.05 seconds, a trivial amount. Since ADIM's capacity is twenty rounds, it is hypothesized that the time needed to update the list is negligible. A thorough simulation of mortars firing and being reloaded is needed to fully test this hypothesis. We plan to make this the focus of our future work.

Acknowledgments/Disclaimer

We would like to thank the entire support staff at USMA's Electronic Support Group and Computer Support Group for their hard work supplying us with hardware and troubleshooting software. We would especially like to thank Mr. Frank Blackmon for assisting in the design of and 3D-printing the Raspberry Pi's cases, Mr. Bob McKay for designing, creating, and fixing the Raspberry Pi's power system, and Mr. Jim Beck for countless hours spent troubleshooting the Raspberry Pi's network and SSH connections. The opinions expressed in this work are those of the authors and do not reflect those of the U.S. Army or the U.S. Military Academy.



Zachary Ramirez is a 2nd Lieutenant in the U.S. Army, Transportation Corps. He graduated with his B.S. in computer science from the United States Military Academy in 2014. He completed the work on this paper as part of an independent study supervised by Dr. Matthews and MAJ Blaine. He is currently stationed in the 916th Sustainment Brigade at Ft. Irwin, CA. He was recently selected to become part of the new Cyber Corps and will make the transition in August 2016.

E-mail: zachary.j.ramirez3.mil@mail.mil



Raymond Blaine was commissioned a Signal Officer and recently became a Cyber Officer. His assignments include a variety of duty positions at Fort Bragg, N. C. He also has served two tours in OIF and one tour in OEF, as a Platoon Leader, Aide-de-Camp to the Chief of Staff MNC-I, and as S6 for 2-508 PIR respectively. He is an Assistant Professor at USMA.

E-mail: raymond.w.blaine.mil@mail.mil



Suzanne J. Matthews is an assistant professor of computer science at the United States Military Academy, West Point. She received her Ph.D. in computer science from Texas A&M University, and her M.S. and B.S. in computer science from Rensselaer Polytechnic Institute. Her honors include a Texas A&M University Dissertation Fellowship, a Rensselaer Master Teaching Fellowship, and memberships in the Upsilon Pi Epsilon and Phi Kappa Phi honor societies.

E-mail: suzanne.matthews@usma.edu

REFERENCES

1. Kowal, Eric and Lopez, Ed. Revolutionary Mortar System to Boost Speed, Accuracy, Enhance soldier Safety [Online]. Available: <http://www.army.mil/article/147037/Revolutionary_mortar_system_to_boost_speed__accuracy__enhance_Soldier_safety/, 2015>.
2. Raspberry Pi Model B+ Data Sheet [Online]. Available: <<https://www.adafruit.com/datasheets/pi-specs.pdf>>, 2014.
3. Gropp, William, Ewing Lusk, and Rajeev Thakur. Using MPI-2: Advanced features of the message-passing interface. MIT press, 1999.
4. Beazley, David. Understanding the Python GIL [Online]. Available: <<http://www.dabeaz.com/python/UnderstandingGIL.pdf>, 2010>.
5. How-To: Turn a Raspberry Pi into a WiFi Router [Online]. Available: <<http://raspberrypi.hq.com/how-to-turn-a-raspberry-pi-into-a-wifi-router/>>
6. Cox, Simon . Steps to Make Raspberry Pi Supercomputer [Online]. Available: <http://www.southampton.ac.uk/~sjc/raspberrypi/pi_supercomputer_southampton_web.pdf, 2013>

Massive Storage in a Miniature (Embedded) Package

Anthony Massa, MNW Tech

Abstract. Data, data is everywhere...and we want ways to get it, store it, transmit it, and mine it. Many different options exist from solid state drives to embedded data recorders. This article takes a look at the fundamentals of an embedded data storage system, the thoughts behind the design decisions and different features to incorporate in an embedded data storage system.

Introduction

The article delves into what data to store, where to store it, and how to get it off the device. I will also get into the considerations of power management in order to extend the battery life of the storage system and ruggedizing the system so that the data is reliable, as well as the different file system options available.

Data Storage System Design

The design of a data storage system has several basic features to consider...and some not so basic. The first question is: What data is going to be stored? Followed by: Where will the data be stored?

In our initial application, the data to store is information that determines what an object is doing as it floated on the ocean's surface. To accomplish this, an inertial measurement unit (IMU) is used to generate the data. It is important to consider other input data sources, typically from various sensors. In this case, having an accommodating platform that can receive analog (via ADC) or digital data is important. For digital inputs, many sensors are designed with widely used serial interfaces such as serial peripheral interface (SPI) or inter-integrated circuit (I2C). Including these serial interfaces in the data storage system design is important to allow easier integration with a wide variety of sensors.

In order to determine where to record the data, the storage speed needs to be determined as well as the capacity for the data. Along with these considerations, the facilities supported by our microcontroller (MCU) are important to consider.

In this case, the MCU included a secure digital host controller (SDHC) interface for SD/MMC/ μ SD cards. Therefore, the first design uses an SD memory card for data storage. This offers the ability to have removable media for data extraction, varying sizes (including large storage space) of storage capacity by simply changing the card, and a compact size.

In a rugged environment using removable media may not be a good idea because of the connector as a potential issue where the card becomes dislodged. There is also the extra cost associated with a connector if price is the key concern.

In rugged environments, soldered flash might be a better alternative such as serial or NAND flash. Considerations of storage capacity and speed still need to be understood in order to ensure the system specifications are met by the hardware. There is also the consideration of how the data is stored to these "hardwired" alternatives where a file system may not accommodate soldered in flash. In that case, a custom driver needs to be developed. Other concerns with soldered flash are wear-leveling and bad-block handling.

Retrieving the Data

After determining the data storage issues, next is to design how the data is extracted from the storage system. Several options exist depending on how the data storage system is going to be deployed. If the system is going to be returned to a lab environment for data extraction and a removable memory card (such as the previously mentioned microSD card is used), most PCs come with interfaces that can accept these types of memory cards directly. This type of interface provides a level of data integrity by reducing the chances of corrupted data during the transfer.

Another option if the system is returned to a lab environment, a serial port such as a UART could be used to send the data for post processing. Easily interfaces to PCs (albeit typically with a USB-to-serial cable) and can include common serial protocols X, Y, or ZMODEM to add a layer of data integrity with packet checksums.

If the system cannot be returned, but has network access, such as an Ethernet connection, then the data can be retrieved using a standard network protocol. This requires the data storage system to incorporate a network stack in order to be able to communicate over the network. A protocol such as the file transfer protocol (FTP) can be used to allow the data to be accessed remotely.

If a wired connection is not possible, then a wireless connection can be designed to get the data off the unit. Wireless connectivity offers a lot of flexibility as far as data removal goes. Several off-the-shelf modules exist that provide the commonly supported (Wi-Fi, Zigbee, Bluetooth) and low power protocols in use today. There are some transmission range concerns depending on how remote the unit is deployed but these types of wireless protocols offer low power capabilities for remote data extraction. Other wireless options are also available including various modem modules such as cellular or satellite. There are also power as well as added cost concerns not only for the modules themselves but also the cost to use the satellite or cellular network to retrieve the data.

Power Management

A key concern for remote systems is power management. If a system is battery powered and needs to last for months or even years without intervention, one of the biggest design considerations is power management. Many MCUs nowadays have several different sleep modes which conserve power. The power management module can then be designed so that the system only wakes up when it is time to collect data samples

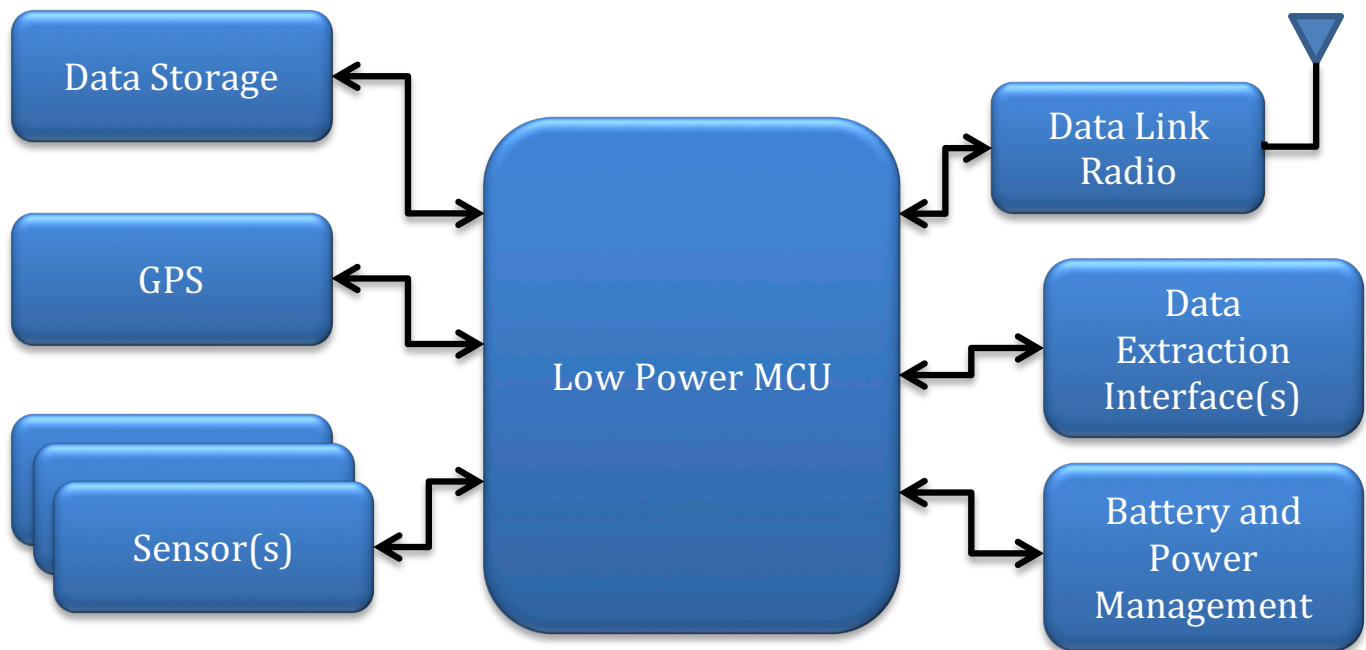


Figure 1. Data storage system block diagram.

from the sensors or time to transmit the data. If a time base is needed to synchronize when data is sampled or transmitted, a GPS module can be included in the system to synchronize time across several units in the network as well as providing location information. A real-time clock (RTC, which can be a module in many MCUs out on the market today or as a standalone chip) can also be used, but needs to be programmed initially with the correct date and time value.

Figure 2 shows the MNW Tech SD card-based data storage system. This system provides a serial interface to extract the data from the system, as well as the capability to remove the SD card directly.

Data Security

In particular applications it may be necessary to secure the data stored to a device for example in medical applications where patient information must be secured. Data security can be a major concern in systems where the storage device is removable and can be tampered with by unknown sources. In these cases the data is encrypted before storing it to the memory device. Various security algorithms exist that provide the necessary security including many open source solutions. In many cases, the microcontroller can assist in the data security by providing a cryptographic acceleration unit. For example, some versions of the Freescale Kinetis ARM-based microcontroller include such a cryptographic accelerator that assists in many different popular cryptographic algorithms including DES, 3DES, AES, MD5, SHA-1, and SHA-256.

System Configuration

In order to develop a general design and product for several different applications, runtime system configuration is needed. Therefore, the data storage system is able to adapt to the specific needs of a particular customer by allowing the customer to select various configuration parameters such as how often the data is stored or data sampling rate, what data is overwritten when the limit is reached, when a new data file is started, and even what data is stored in the file.

There are several different options for runtime system configuration such as a loadable text file which can be contained on the memory card, if present, or downloaded serially. The user can then customize the various data storage parameters by hand-editing the system configuration file. Once deployed, it can be difficult to modify the configuration file unless remote access, such as via wireless download, is available on the system.

Another method which is more elegant is a web interface providing the configuration information. In this case, a network stack and web server need to be incorporated into the data storage system and the customized web page interface needs to be developed. Using a web interface is common on many products today for example on nearly all routers. The user is then able to select the various configuration parameters from drop down lists and radio buttons to customize the operation of the data storage system.

File System

To use a file system, or not...that is a question. For certain,

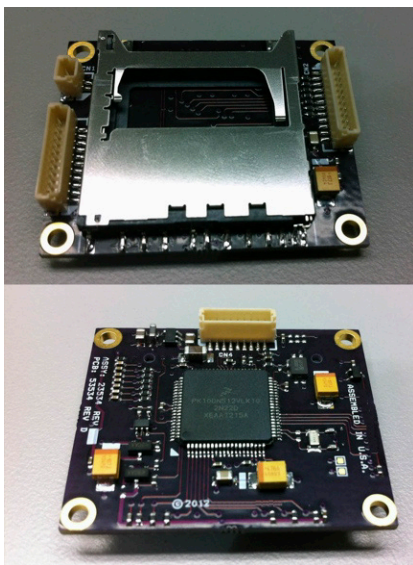


Figure 2. MNW Tech SD card-based data storage system.

more basic data storage applications, a file system can be unnecessary and instead directly writing to the storage module can be used. For example, if a serial flash is used to store the data gathered, only the serial flash driver need be developed and the data can be written directly to the device at the specified addresses. This eliminates layers of complexity with a file system and file management.

In other applications, a file system provides the additional features and capabilities that are necessary. A file system provides help with numerous features including organizing data through the usage of files, timestamp on file data, as well as wear-leveling and bad block management.

Most memory devices have a finite number of program-erase cycles that they are specified to meet. In order to extend the life of the storage device, different memory sectors are used to store the data, essentially mapping the data to different sectors rather than repeatedly storing data to the same sector location in the device. This technique is called wear-leveling. Another technique called bad block management verifies data written to the storage device and keeps track of sectors that are damaged and therefore unusable to avoid data loss.

Several open source file systems are available for use. One popular file system used is FatFS (http://elm-chan.org/fsw/ff/00index_e.html). FatFS is ideal for many embedded systems because it has a small footprint, is written in C making the source code platform independent, and has a very modular design allowing different configuration options. Many of the basic API calls are provided such as open, close, read, write, and mount.

Conclusion

Many different applications exist that require data storage systems from collecting the body's physiological characteristics data to monitoring the temperature of a remote location to tracking the path of an unmanned vehicle. This article has presented a number of key design considerations for an embedded data storage system along with possible solutions based on the type of application in which the data storage system is deployed.

ABOUT THE AUTHOR



Anthony Massa is the Director of Software Engineering at MNW Tech <<https://www.mnwtech.com>> in San Diego, CA. Anthony has over 20 years of experience in all aspects of engineering, focusing on embedded systems and has worked on several successful products. He has taught courses on embedded software development and written extensively on the subject including the books *Programming Embedded Systems: with C and GNU Development Tools* (O'Reilly) and *Embedded Software Development with eCos* (Prentice Hall PTR) as well as several articles.

Phone: 619-252-7010

E-mail: amassa@san.rr.com

International Partners with Multi-Site Thin Client Interconnectivity

Brendan Conboy, Raytheon

Abstract. Do you remember when maintaining a code base with a foreign customer with development and test in both countries meant costly travel, restless nights due to hotel beds and time changes? Well we found a partial solution to this. Imagine being able to work on not only the code base, but virtually in the lab of your foreign customer without having to get on a plane, bus or ship, and enjoying the comfort of your own bed at night. With thin client and VPN technology we were able to save on time and travel costs and be able to more tightly integrate software and system changes. Now we still have language and culture boundaries, but those are out of scope with this article. With the ever evolving virtualization technologies the possibilities are multiplying daily.

Our story starts with an engineer that sees an opportunity to help his colleagues with work/life balance by allowing them to work from home easier with a thin client while tending to a sick child or watching a plumber fix their sink. This led to the idea, well if it works over the internet locally, why shouldn't it work internationally? Well after a long struggle with IT, firewall rules were created and properly tuned to allow the thin clients to exist inside our walls but not necessarily networked internally to maintain security. Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, we can sit at our own desks, make a couple phone calls and connect in to another network. This even allows access to the development and integration labs allowing more efficient debugging with actual live site data.

Can software development be considered tactical? Does the development have an objective, or at least those developing it? Is there a schedule the software development needs to meet? Can a jointly developed software code base between two companies happen in "real time?" Perhaps not completely, however when the code base affects air traffic control and budgets and schedules are tight, it may be closer to those (real time and tactical) than not. When this program started in the very early 1990's the concept of a shared code base that could be developed by both parties in near real time was not even a thought. The best solution at the time was for customer engineers to travel to the contractor's site for initial requirements development and software architecture. Then over time software source code and executables were delivered on tape by hand. Once the system was accepted by the customer and the software warrantee period expired, then all software development moved to the customer's site. This is as far from real time tactical as we get in this article.

About a decade later this software development relationship took a big step forward. With both sides using what is now

IBM's ClearCase software configuration management software, a multi-site relationship was established for a follow on contract and system upgrade. Code updates (what IBM calls synchronization [synch] packets) were encrypted using Gnu Privacy Guard (GPG) software and then exchanged using a now retired File Transfer Protocol (FTP) dropbox. These packets were eventually exchanged every week day via cron tasks on both sides. (Timing was an interesting adventure due to differing time zones and daylight saving time schedules) Now that the code base is synchronized in near real time, what about human communication. Well that is a little harder without a common language; at least the code was all in C. Even though both sides spoke English communication had troubles at times and still a lot of travel was required to either truly test or truly understand new features or integration issues, especially those "you have to see this to believe it..." kind of problems.

Now to introduce the thin client part of the story. The customer site had migrated to a thin client infrastructure for software development. This meant thin clients on developer's desks as well as in the lab for integration and debugging. By default, the thin clients used broadcast traffic to find a server and provide a "window" onto the network. What worked incredibly well as the "window" the developer had at their desk was the same view in the lab. Now take that a little further and consider the situation where the developer is at home either due an appointment or a sick spouse or child. Wouldn't it be great to have that same "window" from home? This would help the developer save valuable vacation time and maintain schedule for the company. So through the use of company supplied SecurID tokens, a corporate VPN concentrator and a version of the thin client's firmware this solution was realized. This network plumbing provides the necessary infrastructure for our ultimate use of thin clients for near real-time development and communication between the two companies.

Both companies were used to the travel and associated paperwork and business went on as usual. Now not everyone has the same schedule nor is available to travel any time or for any length of time. Also in addition to software development the customer started leveraging off the contractor's knowledge of Unix, networking, and development environment infrastructure. This knowledge is best leveraged if one was located at the customer site. Since that wasn't a possibility another solution had to arise, right? This led to the idea "well if the thin clients can connect over the internet domestically, why shouldn't it be able to connect internationally?" One huge difference was that customer's employees already had a SecurID token to use. How do we issue tokens to foreign contractors? Also is it worth opening this development network up to the world; the development network that affects Air Traffic Control?

“Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, the contractor engineers can sit at their own desks, make a couple phone calls and connect in to the customer’s network.”

Now the customer had some work to do. They had to figure out how to allow this access and still maintain its own security. So extra interfaces were added off the thin client servers and connected to a special VLAN. This allows not only special rules to be applied to these thin clients but also an easy way to segregate or disconnect just those thin clients coming from outside the customer’s site. This VLAN runs through the customer’s intranet bound for a VPN concentrator off their firewall. Then a single VPN account for the contractor was created that is shared and linked to a single RSA token that is carried by the “on call” technicians at the customer site. The process for the contractor to connect was to power on the thin client, enter the shared username and PIN, then call the “on call” technician to get the token output. This provides a simple way to connect and preserves security for the customer, as they are the final gate to access. This was later expanded to a secondary shared account, PIN and token that is held by the customer’s network support center that operates 24/7 to better accommodate the time differences between the customer and contractor.

At the contractor site an isolated DMZ VLAN off the corporate firewall was created to house these thin clients. Since there is no permanent storage (besides some flash memory for the basic configuration of the device itself) in the thin client, security was a lot easier to maintain. Since nowadays VLANs can be extended to where ever one needs them. Thin clients were strategically deployed to the contractor’s lab, users’ desks and even common areas to accommodate turnover and even customer visits. Even customer visitors take for granted having their own desktop when they were at the contractor’s site.

Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, the contractor engineers can sit at their own desks, make a couple phone calls and connect in to the customer’s network. This even allows access to the development and integration labs allowing for more efficient debugging with actual live site data. This was made available with the introduction and advancements in virtual frame buffer technologies (VNC and variants). Now while still at the contractor site one can log into the various test strings (even book the test strings like a customer employee) load them with your software and see the output, even interact with them, with live site data not available at the contractor’s site. This is an incredible advancement since the customer site has test strings that emulate the depth and breadth of the fielded systems and includes actual live data feeds and other data inputs. This is a stark contrast to the contractor’s site which has a bare bones system with a minimal subset of the equipment and only test data generators that simulate the live site data.

So from manual hand carrying media on planes and sleeping in hotels to staying home and getting to see the kids’ sports games. We have come a long way in improving not only the speed and agility of our shared software development program. Not only saving schedule and travel expenses, but also helping balance out employees’ lives. Will this work in every situation? You may be surprised. I was a part of it and cannot believe what we were able to do with as little effort as we needed.

Disclaimer:

Copyright © 2015 Raytheon Company. All rights reserved.

ABOUT THE AUTHOR



Brendan Conboy is a Unix systems administrator at Raytheon Company in Marlborough, MA. He supports many programs and wears even more hats. Brendan joined Raytheon 1997 most of that time in Marlborough, but a couple years in St. Petersburg, FL. He is mostly supporting Air Traffic Control programs specializing in OS automation.

Special thanks to Einar Skagen of Avinor, AS in Norway who was the real architect and drive behind this change.

E-mail:

Brendan_E_Conboy@raytheon.com

Threat Modeling for Aviation Computer Security

Abraham O. Baquero, SMRT Software Corporation
Andrew J. Kornecki, Embry Riddle Aeronautical University
Janusz Zalewski, Florida Gulf Coast University

Abstract. The safety of aircraft cannot be analyzed anymore based only on potential hazards and failures. Due to their increasing interconnectivity, modern computer systems are exposed to a variety of security threats. Additionally, complexity of the system may be a source of vulnerabilities opening the system to malicious actions with ultimate impact on safety. Threat Modeling is the technique that assists software engineers to identify and document potential security threats associated with a software product, providing development teams a systematic way of discovering strengths and weaknesses in their software applications. Microsoft's SDL Threat Modeling Tool offers automated analysis of security threats of systems that can be represented using data flow diagrams. The article discusses issues of security in aviation and presents a case study of a realistic cyber-physical system to introduce tool-supported threat modeling method which can be used for unmanned aerial systems security analyses.

Introduction

The quality of a computer system is a combination of several of its properties. For dependable and regulated systems such properties as reliability and safety used to be the primary drivers of quality. Currently, with increasing systems interconnectivity, two additional major properties come into play: security and privacy.

As early as 2006, a study of Common Vulnerabilities and Exposures in software design between 2002 and 2004 revealed a total of "3,595 security bugs from all corners of the software industry [1]." This study demonstrated a wide range of security, privacy, and reliability problems that may affect the overall quality of the software products.

Unfortunately, today's modern applications due to their increasing interconnectivity, are even in greater risk of being exposed to threats than systems involved in the 2006 study. In newer reports, whether by academia [2, 3, 4], government [5] or industry [6], the numbers are even more alarming. Tight development deadlines and the application of rapid design practices often open the possibility of adding flaws to these critical components of a software application.

To minimize, and possibly eliminate, potential threats to security, reliability, and privacy of the product, it is important for development teams to create threat models to determine the highest threat levels and risks involved in all aspects of the software being developed. This essentially requires a formal security-based analysis to predict and mitigate possible attacks on the software.

In the past, aviation systems were effectively protected by rigorous separation from any external access. Proliferation of

modern interfaces and excessive access to the Internet caused that safety of the aircraft cannot be analyzed anymore based only on potential hazards and failures. One needs to consider intended malicious actions that pose security threats with ultimate impact on safety. Therefore, there is a need for guidance identifying methods, techniques, and considerations for securing airworthiness during the aircraft development life cycle from project initiation until the Aircraft Type Certificate is issued.

The methods, techniques, and considerations should address the acceptability of the airworthiness security risk and the design and verification of the airworthiness security attributes as related to system safety. From a perspective of the "C-I-A triad" (Confidentiality – Integrity – Availability), the interest of the guidance are issues of integrity and availability due to their potential impact on safety.

Practices for airworthiness security are undergoing evolution and refinement as new features are deployed and the security threats advance. One of the elements of the airworthiness security process is threat modeling. It is thus critical to explore methods that would facilitate analysis and provide tools to conduct threat modeling.

The objective of this article is to present the issues of security in aviation and to introduce threat modeling as a method to identify security threats. The next sections provide an overview of basic issues and concepts of aviation security, followed by a description of a case study to which threat modeling is applied.

Aviation Systems Security Guidelines

A well-established guidance for aviation software aspects of system certification DO-178C [7], in section 2.3.3, defines the software level matching the system Development Assurance Levels (DAL) defined earlier in section 5.2.1 of ARP4754A "Guidelines for Development of Civil Aircraft and Systems [8]." Software level is pre-determined by the system safety assessment as described in ARP-4761 [9].

Neither of existing aircraft system safety guidance documents does specifically address airborne network and data security issues, which results in non-standardized and potentially inequitable agreements between the various applicants and the regulatory agencies on an acceptable process and means of compliance for ensuring safe, secure and efficient aircraft network design and operations.

Since historically the aircraft systems were deemed protected from external access, DO-178C focus is entirely on safety, neglecting any reference to security. Due to increasing interconnectivity of modern systems and proliferation of unmanned aerial systems, the issues of security became more important. The RTCA and EUROCAE established a new committee SC216/WG71 dedicated to aviation security. In 2014 the committee finalized updated version of "Airworthiness Security Process Specification" DO-326A/ED202A [10] and a new document "Airworthiness Security Methods and Considerations" DO-356/ED203 dedicated to aviation systems security (available at the RTCA site www.rtca.org). The former is a development process standard that the developers must adhere to show that technical requirements are sufficient and they are implemented correctly.

The above mentioned RTCA documents, address the informa-

tion security of airborne systems and related ground systems. They are applicable also to unmanned aircraft which heavily depend on communication for their safe operations. The following discussion is conducted in compliance with current versions of the above documents.

Aviation Systems Security Concepts

We define an asset as a physical or logical resource of the system (functions, subsystems, interfaces, data, processes, and other items valuable to system's operation) which may be subject to attack. In the aviation world, examples of assets include flight deck, flight control, maintenance, navigation databases, navigation services, aircraft software and hardware, etc. The changes in the condition of the assets caused by the attack may have impact on safety. These changed conditions are the threat conditions.

Figure 1 presents a useful high-level model explaining the conceptual relationship between safety and security [11]. Given a system composed of a controller and a controlled process, disturbance in the environment may constitute a hazard to the controlled process of the system operation. The controlled process, on the other hand may be a source of system induced hazards to the environment. Moreover, external conditions be it malevolent attacks or interconnectivity problems can be treated as a threat to the controller being the major system asset. Controller interfaces constitute the potential attack surface and should be treated as trust boundaries.

A security threat may result in a condition with an adverse effect on the system safety involving not only assets but also people or processes – having an adverse effect on the aircraft and its occupants. In conjunction with operating conditions, failure conditions, and environmental conditions, such effect can be either direct or consequential. Threat condition is similar to failure condition, except that the former results from malicious information security attack, while the latter is essentially a hazard resulting from system faults and non-malevolent environmental events.

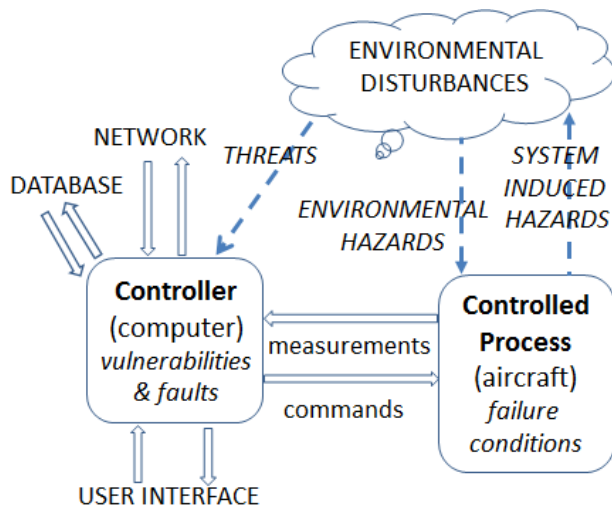


Fig.1. High-level model: controlled process – controller.

Threat condition can lead to exploiting vulnerabilities, where vulnerability is identified as weakness in an information system, security procedures, internal controls, or implementation that could be exploited by a threat source [12]. Vulnerability can be unintentionally triggered or intentionally exploited and result in a violation of the system's security policy.

Threat conditions, categorized as a loss of a security service for an asset (e.g., integrity, availability, or confidentiality), would have impact on safety. Therefore, the SC-216 committee guidance [10] allows to apply the ARP4754A categorization to security. The severity classification of an asset lists its threat conditions and the severity of their impact. Under this assumption, severity of system asset threat conditions can be classified into five categories (I to V) from the highest "catastrophic consequences" leading to the loss of the aircraft to the lowest "no safety effects." Trustworthiness level classifies the trustworthiness according to the impact level of misuse of the assessment asset. The five levels indicate how trustworthy it is to use or manage assets with safety impacts, ranging from none (general) to catastrophic (special trust).

Considering both the severity of the asset misuse by the population and the trustworthiness level, one can determine a likelihood of the asset misuse by trustworthy population, ranging from frequent (anticipated to occur routinely, for no effect and no trust) to extremely improbable (not anticipated to occur, for critical effects and special trust). The likelihood, together with the threat scenario impact, constitutes the risk level matrix identifying acceptable or unacceptable risk.

The starting point in risk analysis is to establish the security perimeter. The questions to be asked are:

- Who can access the system?
- How the system can be accessed?
- Are there any other means to access the system?
- What are specific elements of the system that can be accessed?

Once the security perimeter is established, one needs to identify the security environment. Here the questions to be asked are:

- Who can attack?
- What can be attacked?
- What needs to be protected?

A response to these questions allows the analyst to identify the specific assets to protect and the attack vectors. Threat modeling is a useful technique to assist in such analyses, by addressing questions similar to those stated above.

Threat Modeling

Threat modeling is the activity that assists software engineers to identify and document potential security threats associated with a software product. It provides development teams a systematic way of discovering strengths and weaknesses in their software applications during the Security Design Lifecycle.

There are a couple of tools that can be used in this process [13], PASTA [14], Trike [15], and Microsoft SDL. The latter tool was selected, because at the time of this work it was the most stable and best followed the software engineering principles. The SDL Threat Modeling Tool offers automated analysis of security

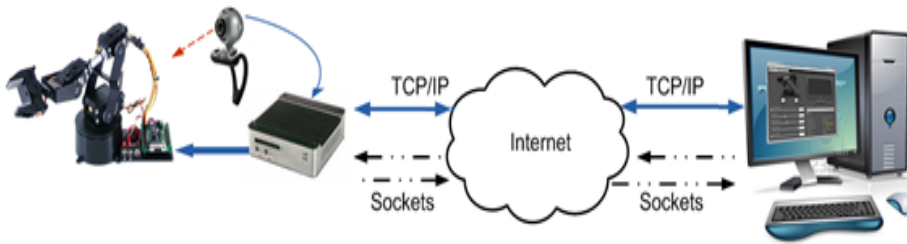


Fig.2. Robotic Application Case Study

threats to a system. It has been described in details elsewhere [1, 16], so here we present only an outline of the technique, referring the interested readers to original publications.

The process starts with building a Data Flow Diagram (DFD) model of the system under analysis. DFD's are a well-established technique used to visualize how the system processes data [17]. Critical in security modeling using DFD's are trust boundaries on paths between the system entities. As defined in the STRIDE model [16, 18], a transition from one trust boundary to another has to validate the following six threat types: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privileges. For example, in aviation environment, spoofing may modify data such as flight plan or GPS data that appears to be from a legitimate source caused by a protocol weaknesses, compromised security data, or compromised ground systems. Tampering uses digital connection to execute malicious instructions on installed equipment. It typically exploits software vulnerabilities such as buffer overflow. Denial of Service uses a digital connection to disrupt service often via inherent protocol features causing flooding or address resolution protocol poisoning. These threats may cause that a malicious content is inserted into a legitimate part, software component, or database causing, e.g., wrong flight approach or automated sabotage.

For an aircraft system, the security risk assessment shall focus on threats and attacks that may affect safety of the aircraft. However, the complexity of most of the aircraft systems is such that it is not easy to assess their safety. The used approach is to identify the conditions that control the airworthiness of the aircraft, and to classify the severity of the impact of all other events in terms of these top-level conditions [8].

The safety analysis considers defects and failures, and a chain of adverse events and conditions causing a potential safety hazard. The effect of a hazard is a failure condition, with an adverse effect on the aircraft, the crew, and the passengers. In a complex system such as an aircraft, one can identify a top-level failure condition composed of combination of item defects or failure conditions along with other operational conditions or events.

A security attack also involves a chain of adverse events and conditions. In this case, by analogy to failure condition, a threat condition will occur through an information security attack. Such system can be presented in form of data flows with appropriate threat boundaries between the processes and thus allowing applying the proposed threat modeling technique.

The analysis would allow designers to apply security measures (deterrent, preventive, detective, corrective, recovery) appropriate to the identified threat. These measures, implemented on threat

boundaries when results of analyses identify the specific threats, can be procedural (procedures, policies, and people) or technical (functions, systems). A case study presented below demonstrates the applicability of threat modeling to reveal some of the threat conditions.

Case Study: System Description

Since modern avionics evolves rapidly towards Unmanned Aerial Systems (UAS), with significant progress in research over the recent years [19, 20, 21] and growing engagement of the FAA [22], the Case Study selected for this project to model threat conditions involves a Remote Robotic Device. It is an example of a remotely controlled cyber-physical system, which can be additionally disturbed by remote alteration (upload) of the control software. A remote user (e.g., pilot, maintenance personnel) can not only issue movement commands to a robotic arm via client application, much like controlling a UAS, but also remotely upload new applications (Figure 2). Additionally, video feedback is provided to monitor the response of the robotic arm to the remote commands imitating the sensor responses notifying about the controlled device status. The device uses the TCP protocol between the server and the client applications. However, another means of connection can be used.

The five essential interfacing software components (applications) are:

- Robot Server – for receiving commands from the remote user subsequently passed to the robotic arm servos.
- Camera Server – for sending video feedback to the remote user.
- Remote Control – on the client computer allowing the remote user to connect and send commands to the robotic arm device while displaying video feedback.
- Update Server and Update Client – to allow a user update the Robot Server software.

The communication between server and client applications supports three essential functionalities:

- CONTROL: The Remote Control client connects to the server using the server IP address and the specified port. The Robot Server application listens for client requests on a specified port controlling robotic arm.
- VIDEO: A visual feedback from the Camera Server to the Remote Control client application on its own specified port.
- UPDATE: Upon establishing a channel of communication between Update Client and Update Server, the replacement version of the software can be transmitted to the server.

From perspective of achieving application security, there are two major objectives:

- How secure is the robot system operation?
- How secure is the process of software update?

In this paper, due to the limited space, we present only analysis of the robot operational security (CONTROL and VIDEO functionalities).

Case Study: System Operation Security Analysis

Given the server is up and running, the process of sending a command to the robotic arm begins by client application first establishing connection with the Camera Server to start the video feedback. The client console is ready to send parameterized command data packets to the Robot Server application conforming to the requirements of the Robot Server application API libraries. The principle of communication is illustrated in a flowchart presented in Figure 3.

The user enters movement commands using the Remote Control client console and the commands are sent to the server. The Robot Server application receives the remote control command parameters and passes them onto the robotic arm API libraries. At this point, the robotic arm executes the commands and the cycle ends. All this takes place while the user on the remote control console monitors the results via the video feedback provided by the Camera Server application.

Two assets are connected during transmission of commands from the client application to the server application; therefore, there is always a possibility of potential threats from attackers wanting to break into this process. Vulnerabilities in the flow of

data from the client to the server and back can open the opportunity for information to be intercepted and modified. When the client computer establishes a connection with the Robot Server application, it must be ensured that the request sent to the server is protected from malicious manipulation. The transmission of commands from the client application has to pass through several trust boundaries, as depicted in the data flow diagram (Figure 4, produced using SDL Threat Modeling Tool).

According to the DFD, the first trust boundary (User Boundary) is passed when the client computer connects to the Robot Server to send movement commands to the robotic arm. Here is where the data packets have to be verified for compliance. The next trust boundary (Process Boundary) is between processes. At this point the robot movement commands received by the Robot Server from the client application have passed the compliance verification and are passed onto the robotic arm library APIs for execution. The final trust boundary (Machine Boundary) defines a transition from the server computer to the physical robotic arm controller responsible for activating the servos to satisfy the movement commands.

Use of Threat Modeling Tool and Analysis of Results

The Microsoft SDL Threat Modeling Tool is a tool to analyze graphically threat models expressed using DFD diagrams. Once the model is created, the tool can analyze it to identify threats and allows the user to enter additional information describing the environment and generate reports.

The Describe Environment feature of the tool helps organize information about the application environment such as the conditions in which the software is deployed, external libraries used by the application (software dependencies), Assumptions of software usage such as how a feature of the software will be used, External Security Notes, and Document Header Information, such as component and product name which appears at the top of all reports.

The tool can be connected to a bug tracking system which allows the user to link a threat to a particular bug in the application. The tool provides also prioritized recommendations for fuzz testing, i.e., feeding of random information into the program inputs. The results of analysis include the status of all the dataflow elements with their threats as found in the threat model. The SDL creates a complete threat model report.

The critical feature of the tool is to display potential threats generated by the DFD and to suggest applicable mitigation strategies. Figure 5 shows the results of the analysis for the DFD from Figure 4. By selecting the potential threat, the tool provides a section where the developer can explain the impact of the threat and select possible solution to mitigate it. A completion bar is displayed when a solution to mitigate the threat is entered by the user (see Figure 5). Additionally, each threat type is followed by some useful suggestions on how to mitigate the threat successfully (Figure 6).

As the result of applying the tool to the Case Study, two common threats have been identified to be the most critical and which

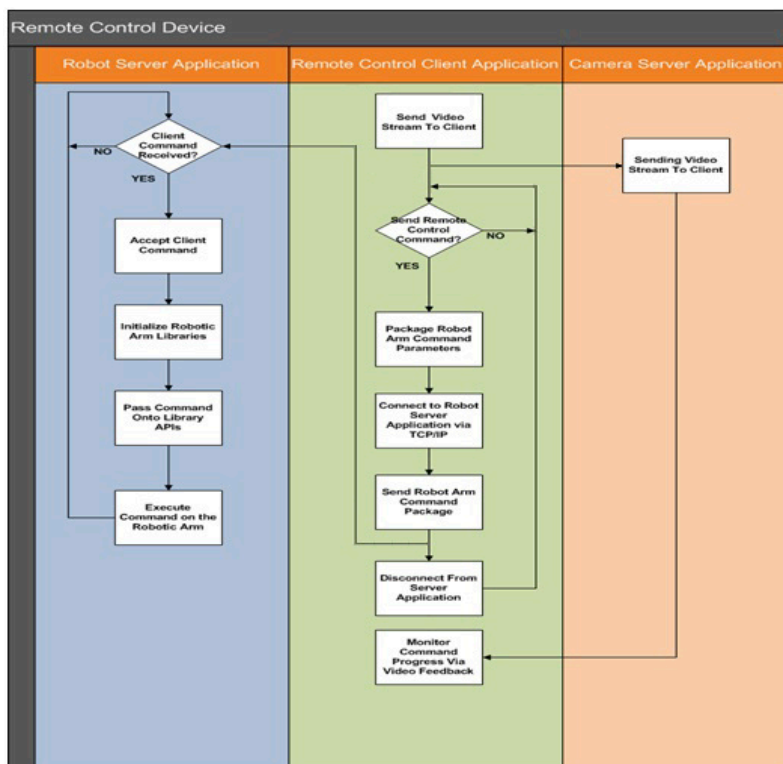


Fig.3. Remote Control Client/Server Interaction

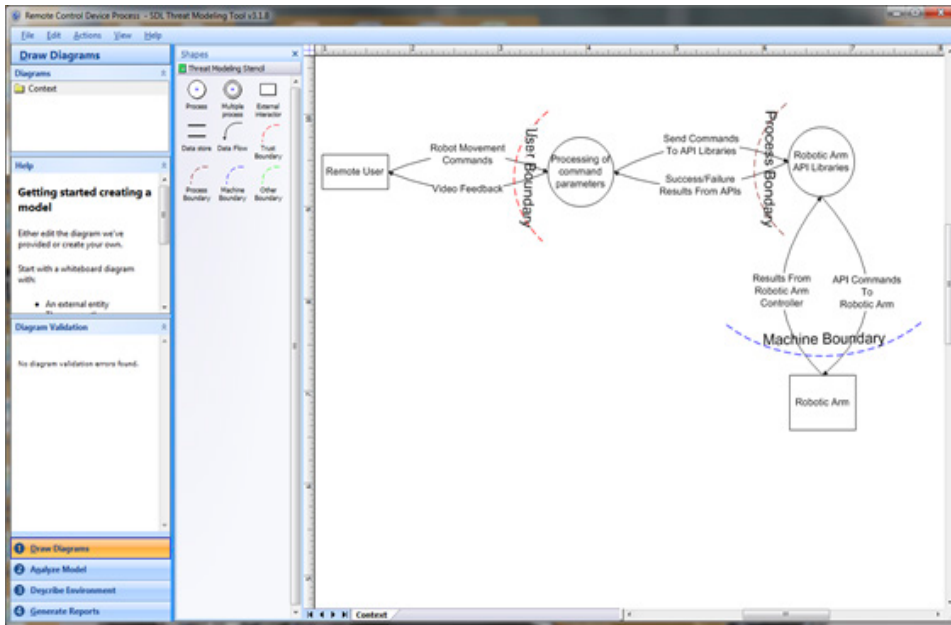


Fig.4. Remote Robotic Control Data Flow Diagram in SDL Threat Modeling Tool

ID	Element Name	Element Type	Element Diagram References	Threat Type	Bug ID	Completion
33	API Commands To Robotic ...	DataFlow	Context	Tampering		
34	API Commands To Robotic ...	DataFlow	Context	InformationDisclosure		
35	API Commands To Robotic ...	DataFlow	Context	DenialOfService		
36	Results From Robotic Arm C...	DataFlow	Context	Tampering		
37	Results From Robotic Arm C...	DataFlow	Context	InformationDisclosure		
38	Results From Robotic Arm C...	DataFlow	Context	DenialOfService		
3	Robot Movement Command...	DataFlow	Context	Tampering		
4	Robot Movement Command...	DataFlow	Context	InformationDisclosure		
5	Robot Movement Command...	DataFlow	Context	DenialOfService		
9	Send Commands To API Libr...	DataFlow	Context	Tampering		
10	Send Commands To API Libr...	DataFlow	Context	InformationDisclosure		
11	Send Commands To API Libr...	DataFlow	Context	DenialOfService		
12	Success/Failure Results From ...	DataFlow	Context	Tampering		
13	Success/Failure Results From ...	DataFlow	Context	InformationDisclosure		
14	Success/Failure Results From ...	DataFlow	Context	DenialOfService		
6	Video Feedback (Processing o...	DataFlow	Context	Tampering		
7	Video Feedback (Processing o...	DataFlow	Context	InformationDisclosure		
8	Video Feedback (Processing o...	DataFlow	Context	DenialOfService		
1	Remote User	Actor	Context	Spoofting		
2	Remote User	Actor	Context	Repudiation		
25	Robotic Arm	Actor	Context	Spoofting		
26	Robotic Arm	Actor	Context	Repudiation		
15	Processing of command para...	Process	Context	Spoofting		
16	Processing of command para...	Process	Context	Tampering		
17	Processing of command para...	Process	Context	Repudiation		
18	Processing of command para...	Process	Context	InformationDisclosure		
19	Processing of command para...	Process	Context	DenialOfService		
20	Processing of command para...	Process	Context	ElevationOfPrivilege		
27	Robotic Arm API Libraries	Process	Context	Spoofting		
28	Robotic Arm API Libraries	Process	Context	Tampering		
29	Robotic Arm API Libraries	Process	Context	Repudiation		
30	Robotic Arm API Libraries	Process	Context	InformationDisclosure		
31	Robotic Arm API Libraries	Process	Context	DenialOfService		
32	Robotic Arm API Libraries	Process	Context	ElevationOfPrivilege		

Fig.5. SDL Potential Threats screen

need to be thoroughly evaluated for software safety and reliability: Tampering and Denial of Service (DoS) attacks.

The first threat, tampering, covers areas where trust boundaries are crossed by the application. It primarily comprises the interaction established by the client computer and the server application when a robot movement command is transmitted. Each of these cross-boundary transitions of data are exposed to external entities that could potentially damage the data flow.

On one hand, there are automated processes capable of reading and modifying bits of information carried in the data packets. A preventive measure is to properly encrypt the information, e.g., by using a 256-bit Rijndael encryption algorithm, to make the information stored in the data packets highly unlikely to decipher by the attacker. Only the server side software with the correct decryption key will effectively decrypt these packets. Such redundant validation of information can be implemented by both, client and server applications to verify the identity of the data to be sent to the server as well as the data received from the client.

On the other hand, to prevent critical information from being exposed to the wrong individuals and potential damage to the system's infrastructure, the authorized personnel need to follow strict protocols to access mission-critical data.

Regarding the second threat, containing a DoS attack is a challenging task since an attacker can continuously change or spoof the location of the IP address from which the attack is incoming so that the source of the attacking process cannot be easily located. Usually, this type of threats are handled by attack-detection tools at the hardware level using firewalls and switches which can be configured to deny traffic from unknown or unusual IP address. While it is unavoidable that respective ports have to be open for the correct functionality of the software, one needs to ensure that only limited number of ports is exposed at any given time and that the software using these ports is up-to-date responding only to valid client requests. This action effectively prevents invalid requests from using valuable system resources.

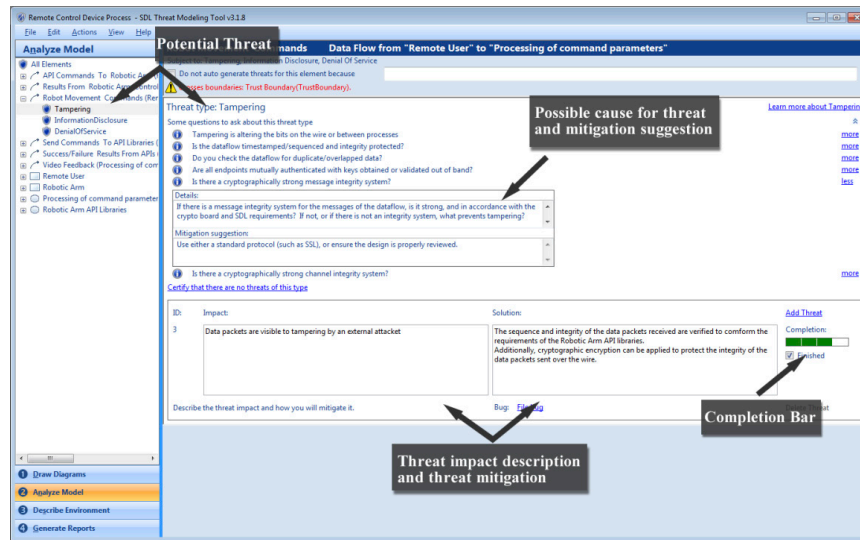


Fig. 6. SDL Potential Threat and Mitigation screen

Conclusion

The use of the SDL Threat Modeling Tool to analyze the Remote Robotic Device infrastructure has proven to be a valuable aid to the early detection and effective prevention of potentially serious flaws in the core of a cyber-physical system with the functionality applicable for computer-intensive aviation system. The presented approach can be used in evaluation of threats for more complex systems as encountered in the aviation industry.

For aircraft system security analysis we start with identifying all functions being considered, determining the severity of the

general failure and threat conditions associated with each function. Subsequently, for all functions the severity of the loss of one of the standard security attributes: Confidentiality, Integrity, and Availability must be determined. Next, for all functions being considered, the severity of the impact of known or obvious attacks including black-box attacks, man-in-the-middle attacks, replay attacks, spoofing and introduction of coherently corrupted messages, and other tampering attacks must be taken into account. The design features or failure conditions which would allow such attacks to be applicable and to succeed are also vulnerabilities that need to be thoughtfully analyzed.

A thorough analysis, following the data flow, allows the identification of vulnerabilities. One needs to consider all functions with data flows or interfaces, physical or logical, to entities that are in a different security domain with a lower security assurance and identify the trust boundaries. Each such data flow represents an inherent vulnerability that could be exploited by an attacker. Special attention must be paid to network layers and non-critical functions which may use operating system layers to manage critical functions. These layers are also exposed and are often based on operating system modules (network stacks, file or memory management, and thread or process management) that may not have been designed for security.

In the context of the work of RTCA/EUROCAE SC-216 committee [10], the threat modeling approach allows developers to address majority of system threats, identify appropriate mitigations on trust boundaries and thus contribute to the improvement of system security. Particularly in aviation applications, threat modeling allows developers to identify the trust boundaries in the security architecture and subsequently apply appropriate mitigation measures on these boundaries. Such approach supports a defense-in-depth concept and through improved security positively affects the safety properties of the final system.

Acknowledgment

Janusz Zalewski would like to thank Steve Drager of Air Force Research Lab in Rome, New York, for inspiring discussions during a Summer Fellowship, which generated the idea for this project.



Homeland Security

The Department of Homeland Security, Office of Cybersecurity and Communications (CS&C) is responsible for enhancing the security, resiliency, and reliability of the Nation's cyber and communications infrastructure and actively engages the public and private sectors as well as international partners to prepare for, prevent, and respond to catastrophic incidents that could degrade or overwhelm these strategic assets. CS&C seeks dynamic individuals to fill critical positions in:

- Cyber Incident Response
- Cyber Risk and Strategic Analysis
- Networks and Systems Engineering
- Computer & Electronic Engineering
- Digital Forensics
- Telecommunications Assurance
- Program Management and Analysis
- Vulnerability Detection and Assessment

To learn more about the DHS, Office of Cybersecurity and Communications, go to www.dhs.gov/cybercareers. To apply for a vacant position please go to www.usajobs.gov or visit us at www.DHS.gov.

REFERENCES

1. M. Howard, S. Lipner, *The Security Development Lifecycle*, Microsoft Press, Redmond, Wash., 2006.
2. P. Meunier, *Classes of Vulnerabilities and Attacks*, In: *Wiley Handbook of Science and Technology for Homeland Security*, J.G. Voeller (Ed.), John Wiley and Sons, New York, 2010.
3. R.A. Gandhi, H. Siy, and Y. Wu, Studying Software Vulnerabilities, *CrossTalk: The Journal of Defense Software Engineering*, Vol. 23, No. 5, pp. 16-20, September/October 2010.
4. L. Bilge and T. Dumitras, Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World. *Proc. CCS'12, ACM Conference on Computer and Communications Security*, Raleigh, NC, October 16-18, 2012.
5. K. Stouffer, J. Falco, and K. Scarfone, *Guide to Industrial Control Systems (ICS) Security*. NIST Special Publication 800-82, National Institute of Standards and Technology, Gaithersburg, MD, May 2013.
6. FireEye Advanced Threat Report: 2013. FireEye Labs, Milpitas, Calif., February 2014. URL: <http://www2.fireeye.com/rs/fireeye/images/fireeye-advanced-threat-report-2013.pdf>
7. DO-178C, *Software Considerations in Airborne Systems and Equipment Certification*, RTCA SC-205, 2011.
8. ARP4754A, *Guidelines for Development of Civil Aircraft and Systems*, Society of Automotive Engineers, August 1995. URL: <http://standards.sae.org/arp4754a/>
9. ARP4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, Society of Automotive Engineers, December 2010. URL: <http://standards.sae.org/arp4761/>
10. DO-326A, *Airworthiness Security Process Specification*, RTCA SC-216, December 2010.
11. A.J. Kornecki, J. Zalewski, *Aviation Software: Safety and Security*. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. Webster (Ed.), John Wiley and Sons, New York, 2015.
12. Committee on National Security Systems, *National Information Assurance Glossary*. CNSS Instruction No. 4009, 26 April 2010. URL: www.ncix.gov/publications/policy/docs/CNSSI_4009.pdf
13. Comparison of Threat Modeling Methodologies, May 28, 2012. URL: <http://myappsecurity.com/comparison-threat-modeling-methodologies/>
14. PASTA Process for Attack Simulation & Threat Assessment, 2013. URL: http://versprite.com/docs/PASTA_Abstract.pdf
15. E. Saïta, B. Larcom, and M. Eddington, *Trike v1: Methodology Document (Draft)*, July 20, 2005. URL: <http://octotrike.org/papers/>
16. F. Swiderski, W. Snyder, *Threat Modeling*, Microsoft Press, Redmond, Washington, 2004.
17. P.T. Ward and S.J. Mellor, *Structured Development for Real-Time Systems*. Vol. 1, Prentice Hall, Englewood Cliffs, NJ, 1985.
18. D. Dillon, *Developer-Driven Threat Modeling Lessons Learned in the Trenches*, IEEE Software, Vol. 9, No. 4, pp. 41-47, July/August 2011.
19. T.G. McGee et al., *Mighty Eagle: The Development and Flight Testing of an Autonomous Robotic Lander Test Bed*. Johns Hopkins APL Technical Digest, Vol. 32, No. 3, pp. 619-635, 2013.
20. R. Black, M. Fletcher, *Simplified Robotics Avionics System: A Integrated Modular Architecture Applied Across a Group of Robotic Elements*, 25th Digital Avionics Systems Conference, 2006 IEEE/AIAA, pp.1,12, 15-19 Oct. 2006, DOI: 10.1109/DASC.2006.313775
21. F. Boniol, V. Wiels, *Towards Modular and Certified Avionics for UAV*, *AerospaceLab Journal*, Issue 8, December 2014. DOI : 10.12762/2014.AL08-02.
22. RTCA Inc., SC-203. DO-344 Volume 1 & 2 - Operational and Functional Requirements and Safety Objectives for Unmanned Aircraft System Standards, June 19, 2013. URL: http://www.rtca.org/store_product.asp?prodid=1113

ABOUT THE AUTHORS



Abraham O. Baquero, B.Sc. in Computer Science, is the Chief Technology Officer and one of the founders of SMRT Software Corporation. His expertise includes software piracy prevention, software licensing and intellectual protection. He held the position of Lead Software Architect designing and implementing e-commerce software platforms. His responsibility was security, availability and stability of the financial processes. Most recently he engaged in analysis of security of cyber- physical systems in cooperation with Florida Gulf Coast University.

422 Wekiva Rapids Dr., Altamonte Springs, FL, 32714
Phone: 239-645-6831 E-mail: abebaquero@gmail.com



Andrew J. Kornecki, Ph.D. '74, MSEE '70, is a professor of software engineering with interest in safety critical software. He was engaged in activities of the Society for Computer Simulation, IFAC Automation TC, the National Academy of Sciences, and the RTCA committees dedicated to aviation software certification. He cooperated with aviation and medical industries and contributed to real-time safety critical software training for the FAA Certification Services. He worked on several FAA contracts related to airborne software and hardware certification.

Electrical, Computer, Software, and System Engineering Department, Embry Riddle Aeronautical University
600 S. Clyde Morris Blvd., Daytona Beach, FL 32114
Phone: 386-226-6455 E-mail: kornecka@erau.edu



Janusz Zalewski, Ph.D. 1979, MSEE 1973, is a professor of software engineering with interest in safety and security of embedded and cyber-physical systems and engineering education. He worked for nuclear research institutions, including Lawrence Livermore Laboratory, and consulted for industry, including Lockheed Martin, Harris, and Boeing. He served as a chairman of the IFIP WG 5.4 on Industrial Software Quality and spent several summers at Air Force Research Labs, studying trustworthiness and security of cyber-physical systems.

Department of Software Engineering, Florida Gulf Coast University, 10501 FGCU Blvd, Ft. Myers, FL 33965
Phone: 239-590-7317 E-mail: zalewski@fgcu.edu

Extending Life Cycle Models for a Repeatable Innovation Strategy

Duffy Nobles, U.S. Dept. of the Treasury
Kevin MacG. Adams, Ph.D., University of Maryland

Abstract. The goal of many organizations is to be recognized as a business leader that consistently delivers innovative products and services. Different types of life cycle models have been used to guide the systems development efforts and implementation processes within these organizations, all with various outcomes. This paper first explores the reasons why innovation is so elusive, so difficult to achieve and almost impossible to predict. It then explores the possibility of enhancing existing life cycle frameworks so that innovation and break-through accomplishments become part of the organizational structure, not just a random or one-time achievement. It also identifies modern examples and other research data to identify such factors as the expansion of knowledge assets, new patterns for collaboration, environments for radical creativity and transformational skill sets. These findings suggest that a life cycle methodology with the necessary attributes can increase the probability for achieving a repeatable process for innovation.

1. Introduction

Today's most dynamic and successful organizations face constant pressure to expand market share with products and services that are attractive to a shifting and sophisticated global population. Organizations, especially those operating in competitive, technology-driven environments must establish strategies that embrace creativity and innovation in order to maintain hard-won reputations for consistently providing exciting and desirable products [1]. Unfortunately, organizations often repeat strategies that proved effective in the past, but find that those old patterns no longer provide the spark captured by systems and products that are considered truly innovative. This leads to the question: can a life cycle process be used to define, capture and be systematically applied to provide businesses with a repeatable format that consistently delivers innovative and cutting edge developments?

2. Architecting Innovation

Innovation is a recognizable element that expands, defines and delivers solutions to both existing and unimagined needs in a novel and effective manner. Innovation differentiates companies by providing an aura of originality and creativity that customers appreciate and competitors tend to imitate. Innovative products and services can influence consumer trends and have the potential to impact markets on a global level. But this achievement is not guaranteed nor can it be predicted with assurance, even with businesses known for past exceptional innovative accomplishments.

There are several factors that make implementing a repeatable process that delivers innovative products difficult. The most dominant is the perception of risk and the uncertainty that is inherent with any new, untried endeavor [14]. The investments in knowledge, time and financial commitments needed to identify and develop untried products require a leap into the unknown that in the end, still could fail to capture customer expectations or fall short of business objectives. Many corporate leaders view the aggressive investments that innovation demands as "a high-risk, high-cost endeavor, that promises uncertain returns" [20] with "challenges [that] often are considered just too high a risk" [14].

Creating a culture of innovation is a commitment that moves the organization beyond the expected modes of thinking and past its current business practices. The decision to be a corporate innovator requires developing the resources and promoting a strategy for generating the new concepts needed for a "radical model that challenges fundamental assumptions" [15].

A workable life cycle radical model for innovation would necessarily provide a usable framework that applies a repeatable and realistic structure lifecycle. A suitable methodology would encourage a system-wide, possibility-oriented approach that would be more conducive for innovative systems and work products. This type of non-linear process would represent a significant departure from the more traditional ends-oriented approach used by most enterprises today [2].

There are many determinants that can prove useful for calculating an organization's level of commitment to systematic innovation. There are, for example, methods and techniques that map the degree of an organization's performance in relation to global trends and technology developments [3]. The potential for establishing a successful innovation environment can actually be estimated by considering the impact on four specific elements: product, process, position and paradigm [3] and the amount of resources and degree of importance that the enterprise applies to each one.

Life cycles often include iterative stages where system capabilities, functional requirements, technical enhancements and design features are periodically updated to keep a product or service competitive and current. Innovation, on the other hand, requires a paradigm shift that results in something entirely original, that is recognized as "something new that didn't exist before" [2]. The term radical innovation describes the acquisition of a truly unique offering or a novel technology that differs dras-

tically from any preexisting alternatives. It requires a different cognitive frame of reference, one that generates new ideas and assumptions and becomes much more than just the introduction of a leading-edge product or a new service or technology [4]. This is achieved by, what may be regarded at first as, a risky commitment to an ambiguous, resource-intensive learning process. Success often results in changes that lead to the displacement of system capabilities and knowledge investments already established by other competitors and major business players [4].

Radical innovation can indeed be disruptive [5]. But, it is also synonymous with ground-breaking, future-focused products which, in turn can become engines for rapid economic growth with the “power to create entire industries” [6] and change the competitive landscape. Clearly, committing organizational resources to the pursuit of innovation can be an extremely uncertain and risky process for a number of reasons [3]. Previous assumptions derived from existing technologies suddenly become irrelevant in that the available existing knowledge and experiences have little value in the context of the new innovation [4]. But upsetting the existing status quo in this manner can also be viewed as a corporate advantage.

3. The Dynamics of Innovation

Can the concept of innovation realistically be deconstructed, analyzed and reapplied by an organization into a repeatable lifecycle process that consistently generates inventive products? Developing a culture dedicated to innovation is the stated goal of many organizations. It is often included in their strategy and mission statements and identified as a technology or system objective. Unfortunately, achieving this goal is unpredictable; few businesses “seem to [understand] the very notion of innovation and how to apply it... innovation is often misunderstood [and] considered too difficult for practice” [2].

Innovation is characterized by the degree that a new system, product or process is developed from new technology and ideas that differ substantially from what existed before [6]. A life cycle that consistently achieves dramatic break-throughs requires structures and processes that create emergent, non-linear improvements on a recurring basis. The real value comes from combining the “knowledge... the direction, the purpose, [and] the focus [toward] innovation” [2]. Such a knowledge-focused model would redefine connections between the acquisition procedures, the application of new tools, changing technology platforms, and ever-rising expectations to expand assumptions and possibilities [4]. These then become the new knowledge assets that establish the organization's ability to “identify, acquire, integrate and exploit” [4] both the practical and intangible elements needed to support a life cycle process that is conducive to an on-going culture of innovation [2].

Innovation defies prediction; if it was predictable, “then it wouldn't be innovation” [2]. Nevertheless, increasing opportunities for innovative activities require that all system resources, components, strategies, etc. collectively form an environment where a higher degree of creative freedom becomes a possibility [2]. Clearly, expansion of creativity would be a major factor of the innovative life cycle methodology, where expecta-

tions become free of the deterministic restrictions of existing system-building assumptions. This would encourage a systematic culture that promotes the kind of corporate mindset that is “necessary to invigorate and regenerate the firm's life” [7]. In this case, that means a dynamic shift toward fostering non-linear learning experiences by “encouraging, recognizing, and rewarding creativity” [1]. These inducements stimulate the long term conditions of generative learning that are needed for “architecting the dynamics of innovation” [2]. In this context, innovation becomes a real possibility.

4. Life Cycle Models

Companies known for reliably delivering products and services that consistently raise the bar for innovation and new advancements are usually considered to be focused and forward thinking as well. Credit for this is usually given to the “free will and creative activity of the [firms] and their decision making” [7] as well as the “know-what, know-why, and know-how” [4] that is strategically encapsulated by the business and product life cycle models.

One common factor that “all systems and models have is that they involve abstractions” [8]. A model for innovation would be no different. Unfortunately, as stated, organizations, like other entities, tend to pursue the same strategies that proved effective in the past. The tried and true organizational structures and knowledge baselines must somehow bend with changing technologies and expectations, if not, they may become dated and ineffective when new stages of development occur. Yesterday's great and admired innovations soon become technological relics of the past as they are unceremoniously discarded for the next new thing.

Many companies have enjoyed such impressive successes with their innovative achievements in the march toward today's modern computing capabilities [19]. Corporate reputations have been preserved over time through aggressive investments in new technologies, but many of these companies are no longer regarded as leaders of innovation. The pioneers in mainframe computing, for example, missed the emergence of the mini-computer. Many minicomputer manufacturers, in turn, failed to capitalize on desktop computer [17].

The rise of mobile computing and social marketing presented additional opportunities for the forward thinking organization. Innovation is now expected. Tracking consumer opinions and influencing acceptance decisions is considered a competitive advantage. This advantage can actually be achieved by the manipulation of a specific set of “innovation attributes” [18]. These attributes include controlling the perception that the product is superior to its rivals, that the product is compatible with personal values and that the complexity associated with understanding and using the product falls within an acceptable range [18].

The goal then is to establish a life cycle methodology where success is defined by continually identifying the key factors needed to transform and restructure the activities that lead to new break-through opportunities [5]. “The life cycle literature is replete with different models” [7] attempting to identify the difference-making factors and then re-composing them into new

Table 1: Meta-system Element and Coordinating Relationships [8]

Meta-System Elements	Coordinating Relations	Description
The Empirical	Committing	Conclusions are drawn from research data
The Actual	Convincing	Information that is bound by the context of its own situation
The Real	Adjusting	Contains the deeper-level regularities of system behavior which drives the other two

sets of rules, phases, iterations or build components. Appropriate models must provide the system guidance and controls throughout all life cycle phases that can be applied to both the business strategies and the changing competitive environment [10]. Successful organizations manage to weave the concept of simultaneously running their business while changing their business at the same time [11]. This ambidextrous behavior applies to “the organization’s strategies, systems, scorecards and incentives” [11].

5. Models and Examples

There are two models that offer elements that are useful for advancing the concept of an innovation life cycle. The first comes from the Theory of Retroduction Abduction where empirical research and pre-existing bodies of abstract ideas are used to develop conceptual models [8]. One model, based on the Software Development Life Cycle (SDLC) identifies three aspects that form a basic framework for synthesizing and formalizing these empirical data into a life cycle meta-system. These elements consist of: the empirical, the actual and the real. Another similar model, based on the same reasoning and SDLC methodologies, was cited as representing a more modern management system that was recently developed and used in China [8]. This model extends the original elements by examining their relational impact on individual life-cycle events regarding the levels of “adjusting, convincing, and committing” [8]. These three relational coordinates link to the three meta-systems elements. They differ in that they are seen as themes that explain the relationships needed for coordinating and adjusting subsequent life cycle stages by acting together to solve any problems that surface. Table 1 compares the meta-system elements with the relational values.

These concepts not only suggest an approach for identifying and embodying the intangible factors and attitudinal shifts needed for sustained creative thinking, they also describe meta-system elements by which innovation methodologies can be developed and operated.

The second example is derived from the Theory of Absorptive Capacity (ACAP) which represents a knowledge-based model for radical innovation [4]. The ACAP model defines an organization’s behavioral routines by demonstrating how effectively it identifies, acquires, integrates, and exploits knowledge relative to the quality of its domain intelligence and access to technologies. The quality of an organization’s total knowledge portfolio is defined by two dimensions, according to: a) “what it knows – its knowledge base” and, b) “what it does – its routines” [4]. This directly corresponds to the dual concept of idea generation and idea implementation [12]. The model is depicted in Figure 1.

This model shows how specific factors affect different types of organizations by focusing on their internal and external knowledge adoptions. These adoptions can occur either separately or in combination, depending on the knowledge base, but when successful, they result in the generation of new knowledge paradigms. In other words, the analytical results reveal the level of innovation potential by simply showing that an outcome “can be defined as the number of...innovations it adopts” [4] through the management and execution of the corporate knowledge assets defined in Table 2.

When traditional publishing firms were faced with the challenges of the emerging, on-demand publishing market, they were forced to develop new methods for delivering specific customer-defined information services, media formats, etc. To succeed, the firms had to restructure the basic knowledge assets already embedded within their data stores and shift into new capabilities and strategies for product development and information dissemination [16]. This absorption process, along with a redefinition of assumptions, allowed data to be reconfigured and transformed into structures that supported the implementation of innovative information products and content delivery systems.

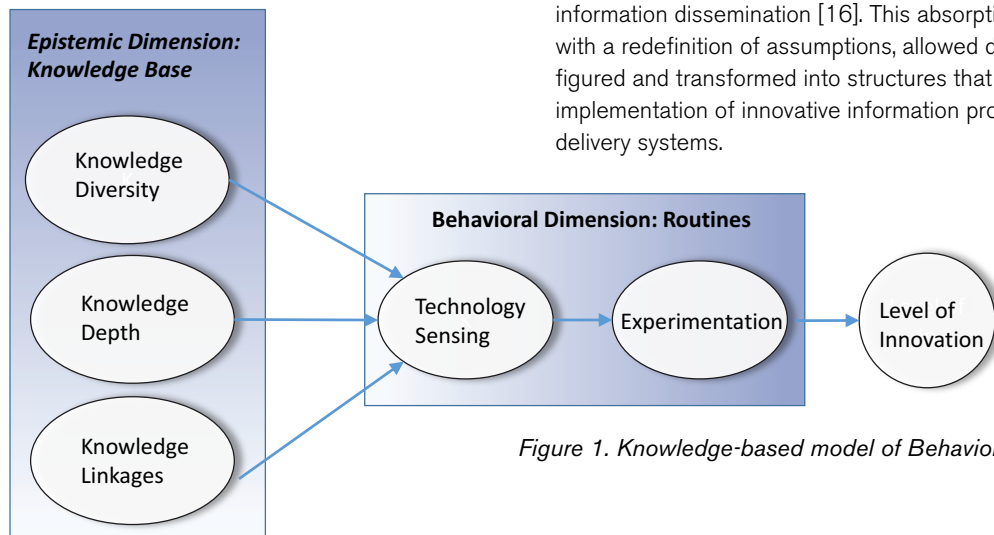


Figure 1. Knowledge-based model of Behavior and Innovation [4]

Table 2: Knowledge Assets and Descriptions [4]

Asset	Description
Knowledge Diversity:	Denotes the extent to which a wide distinction of unique knowledge elements influence specific tasks
Knowledge Depth:	Represents the level of detail, knowledge quality, and depth of domain expertise that can be leveraged.
Knowledge Linkage:	Refers to the channels through which gathered and accumulated, the relationships formed with vendors, clients, scholars and experts.

When included as building blocks in life cycle strategies, these factors encourage innovation by increasing cooperative creativity, establishing non-linear channels for thoughtful collaborations, and allowing for the possibility of serendipity, etc. This kind of business intelligence, in turn has a direct and long-term impact on the organizational behaviors and decision-making responses when endorsed in the life cycle methodology.

6. Transformational Leadership

Repeatable innovation requires that an organization's life cycle methodology has the built-in foresight and flexibility, on a systems level, to adjust to changes in the direction of technological discoveries, the competitive environment, the business mission, and to product or service outcomes [10]. Ultimately, however, it is the capabilities demonstrated by talented leadership that plays an integral part in the process by setting the high expectations and directing the activities needed to sustain environments that favor creativity and innovation [13].

It was found that skilled transformational leaders actually enhance the probability for innovative outcomes [1]. They serve as critical influences for overcoming organizational and team hurdles by providing a specific and recognizable transformational leadership style.

Transformational leaders display a certain degree of behaviors that emphasize change, encourage out-of-the-box thinking and promote individual empowerment [12]. They accomplish this in two ways. First, by constructing a creative environment that favors innovation and second, by directing the strategic goals, activities and expectations needed to sustain the performance of project members and other contributors at high levels [13].

Leaders display transformational abilities by articulate a compelling and inspirational vision. They raise the confidence, aspirations and performance expectations of their followers [1]. This lends proof to the suggestion that "leadership is among the most important factors affecting innovation" [1]. The transformational leader establishes a creative knowledge environment or CKE which ensures that "the social and organizational characteristics at the team and organizational levels, have a crucial influence on the innovation processes" [13]. They motivate the team with an attractive vision of future states.

Transformational leaders are skilled at motivating people and their leadership style convinces teams to buy into their visions and work ethics by steering workplace perceptions in ways that influence and encourage the desirable innovation behaviors [1]. Quality team performance is defined as the quantity of implemented ideas "in terms of [their] novelty, magnitude, radicalness and effectiveness"; in other words, the degree of originality and inventiveness; the characteristics of innovation [12].

7. Conclusion

Many organizations are capable of producing a system, a product or a service that is considered radically innovative, but repeating the feat is often elusive and in many cases impossible. New technology developments combined with shifting collaboration patterns can expand the corporate knowledge base and create new possibilities that never existed before. The factors that contribute to creative, non-linear, out of the box thinking when identified and isolated can be decomposed and reassembled into an enhanced life cycle methodology where innovation becomes a repeatable part of the development process. Innovation can never be predicted, of course. But with focused, transformational leadership, a solid understanding of necessary system meta-elements, and an environment that encourages radical creativity, a foundation can be developed where innovation is expected and can be sustained.

So, to answer the question posed in the introduction: can a repeatable life cycle be defined and applied as a business process that consistently delivers innovative and cutting edge systems and products? Well, the answer needs much more research, but the foundational elements for such a consideration certainly already exists.

ABOUT THE AUTHORS



Duffy Nobles, PMP, ITILv3, has years of experience leading highly performing teams in planning, managing and developing solutions for complex federal and commercial systems. Until recently, he worked for the MITRE Corp. where he provided systems engineering services for projects at the U.S. Census Bureau, the Transportation Security Administration (TSA) and the Department of Veterans Affairs (VA). He is currently managing international compliance and taxation projects at the U.S. Dept. of the Treasury.

**9017 Gettysburg Lane
College Park, MD 20781
Phone: (301) 219-7499
Email: dufnobl@hotmai.com**



Dr. Kevin MacG. Adams is an Adjunct Professor at the University of Maryland University College where he teaches software and systems engineering in the graduate program in Information Technology. Dr. Adams is a retired Navy submarine officer and information systems consultant. Dr. Adams holds a B.S. in Ceramic Engineering from Rutgers University, an M.S. in Naval Architecture and Marine Engineering and an M.S. in Materials Engineering both from MIT, and a Ph.D. in Systems Engineering from Old Dominion University.

**University of Maryland University College
3501 University Blvd. East,
Adelphi, Maryland 20783
Phone: (757) 855-1954
Email: kevin.adams@faculty.umuc.edu**

REFERENCES

1. L. Gumusluoglu and A. Ilsev, "Transformational Leadership and Organizational Innovation: The Roles of Internal and External Support for Innovation*," *Journal of Product Innovation Management*, vol. 26, pp. 264-277, 2009.
2. T. Jörg and S. Akkaoui Hughes, "Architecting the dynamics of innovation," in *Proceedings of the International Conference on Intellectual Capital, Knowledge Management & Organizational Learning*, L. Garcia, A. Rodriguez-Castellanos, and J. Barrutia-Guenaga, Eds., ed. Reading, UK: Academic Conferences and Publishing International Limited, 2013, pp. 222-230.
3. R. H. Abdel-Razek and D. S. Alsanad, "Mapping Technological Innovation: Methodology and Implementation," *Global Conference on Business & Finance Proceedings*, vol. 8, pp. 175-183, 2013.
4. J. L. Carlo, K. Lyytinen, and G. M. Rose, "A knowledge-based model of radical innovation in small software firms," *MIS Quarterly*, vol. 36, pp. 865-896, 2012.
5. J. P. Dismukes, J. A. Bers, and J. A. Sekhar, "Toward a holistic six-period radical innovation life cycle model," *International Journal of Innovation & Technology Management*, vol. 9, pp. 12500011-125000127, 2012.
6. V. Eiriz, A. N. A. Faria, and N. Barbosa, "Firm growth and innovation: Towards a typology of innovation strategy," *Innovation: Management, Policy & Practice*, vol. 15, pp. 97-111, 2013.
7. D. L. Lester, J. A. Parnell, and M. L. Menefee, "Organizational life cycle and innovation among entrepreneurial enterprises," *Journal of Small Business Strategy*, vol. 19, pp. 37-49, 2009.
8. C. M. Brugha, "Implications from Decision Science for the Systems Development Life Cycle in Information Systems," *Information Systems Frontiers*, vol. 3, pp. 91-105, 2001.
9. L. Greiner, "Evolution and revolution as organizations grow," *Harvard Business Review*, vol. 50, pp. 37-46, 1972.
10. S. M. Auzair, "Organisational life cycle stages and management control systems in service organisations," *International Journal of Business and Management*, vol. 5, pp. 56-65, 2010.
11. J. R. Latham, "Leadership for Quality and Innovation: Challenges, Theories, and a Framework for Future Research," *Quality Management Journal*, vol. 21, pp. 11-15, 2014.
12. S. A. Eisenbeifs and S. Boerner, "Transformational Leadership and R&D Innovation: Taking a Curvilinear Approach," *Creativity and Innovation Management*, vol. 19, pp. 364-372, 2010.
13. L. Denti and S. Hemlin, "Leadership and innovation in organization: A systematic review of factors that mediate or moderate the relationship," *International Journal of Innovation Management*, vol. 16, pp. 12400071-124000720, 2012.
14. M. Johnson, "Barriers to Innovation Adoption: A Study of e-Markets," *Industrial Management and Data Systems*, vol. 110, pp. 157-174, 2010.
15. J. Euchner, "The Uses and Risks of Open Innovation," *Research Technology Management*, vol. 56, pp. 49-54, 2013.
16. F. A. Van, H.W. Volberda and M. de Boer, M, "Coevolution of Firm Absorptive Capacity and Knowledge Environment: Organizational Forms and Combinative Capabilities," *Organization Science*, vol.10, pp. 551-568, 1999.
17. C. M. Christensen and J. L. Bower, "Customer Power, Strategic Investment, and the Failure of Leading Firms," *Strategic Management Journal*, vol. 17, pp. 197-218, 1996.
18. G. Roach, "Customer Perceptions of Mobile Phone Marketing: A Direct Marketing Innovation," *Direct Marketing*, vol. 3, pp. 124-138, 2009.
19. N. Ensmenger, "The Digital Construction of Technology: Rethinking the History of Computers in Society," *Technology and Culture*, vol. 53, pp. 753-776, 2012.
20. T. D. Kuczmarski, "What is Innovation? The Art of Welcoming Risks," *The Journal of COnsumer Marketing*, vol. 13.5, pp. 7-11, 1996.

Mobile and Embedded Security Mitigations for Counterfeit Threats and Software Vulnerabilities

Jon Hagar, Grand Software Testing

Abstract. Mobile and embedded software teams, users and stakeholders have historically underestimated the risk of security threats. As a result, vulnerabilities that can be exploited by hackers to gain access to mobile and embedded software devices are on the rise and will only continue unless the programming and testing staff takes measures to prevent them. These vulnerabilities can come from many sources such as: inadequate architecture or poor design, software coding errors, and intentional code such as viruses and back doors inserted into the code during development or updates. This paper examines a variety of good engineering practices that should be considered to minimize and control vulnerabilities during development and test activities. While much of this advice may seem common to historic information technology (IT) security concepts, it is still under used in many mobile and embedded system projects. An overview of testing attack concepts with specific considerations for mobile and embedded software devices will be introduced.

Introduction

Software has had security issues since its inception. Now with nearly everyone on the planet carrying networked computers in their pockets added with the rise in hacking (gaining unauthorized access), vulnerabilities must be prevented. With the increased use of embedded devices and mobile systems (smart phones, pacemakers, cpus and interface units in cars and Bluetooth networks in and connected to cars, tanks, Humvees, etc.), devices that once were not considered a security risk should always be considered vulnerable. The industry is experiencing hacking of embedded systems such as avionics (GPS spoofing of drones, even cars), medical (pacemakers), and factory systems (controllers). Soldiers now carry "personal" cell phones as well as other "smart" devices into the battlefield to assist them in tracking friendlies and enemies" or to call in air strikes, as simple examples. These example mobile and embedded systems and the software executing on them represent non-traditional threats yet are examples of areas where software compromises must be stopped.

It may be tempting to consider mobile and embedded devices using the same security measures as traditional IT systems. In some cases, this may mitigate vulnerabilities. However, mobile

and embedded devices have features that set them apart from traditional IT systems including:

- Networking situations that change quickly or networks that can be subject to vulnerabilities themselves;
- Mobility issues such as device ownership that can be changed or lost without prior knowledge or devices with features supporting movement, etc.;
- Resource limitations such as power, processing speed, memory, and certain timing issues;
- Environmental factors such as water, sun, cold, heat, dust, moisture, and so forth;
- Different user interfaces (UIs) such as smaller screen size, readability, touch screen issues including swiping;
- Operability on many hardware platforms and integration issues;
- Different cycles of software and hardware updates and how those are handled, compared to the classic IT domain.

Mobile and embedded development teams have begun to recognize the impact of these aspects but are slowly reacting to the security threats. Further, mobile and embedded devices may contain unknown, even counterfeit software that can present a threat without the knowledge of the user or network. Even without counterfeit software, the system may come with vulnerabilities [1, 2]. Further, the rapid growth of 25% (or more) per year in mobile and embedded software markets [3] means that security risks are likely to increase.

This paper examines software lifecycle considerations for reducing security threats with a focus on security test attacks. Since security topics are large and complex only basic introductions are presented with pointers to further information.

Software is now embedded in cars which are being hacked [4, 5], industrial control systems which are at risk [6], mobile devices with vulnerabilities [7, 8, 9, 10], medical devices that have been hacked [11] and other areas, all of which face security concerns. The paper defines development mitigations including architecture, design and coding practices. Then, security attacks that the test team should consider are summarized. While a majority of the concepts presented are applicable to various types of software domains, many of the concepts presented are either not well known or well-practiced within mobile and embedded software development projects, as illustrated in error taxono-

mies reported in [2, 5, 7]. A goal of this paper is to increase the awareness on mobile and embedded development projects of basic security concepts, while presenting some security concepts tailored to mobile and embedded domains.

The Situation: Mobile and Embedded Software Security Threats [1]

At one time, not too many years ago, mobile systems, did not exist and, embedded software devices were safely sequestered and isolated from networks. Thus, security threats to such devices and systems were limited or not considered, at all. Mobile and embedded development teams may have introduced security vulnerabilities but did not conduct security testing because of the lack of risk. This is changing. For example, a Coverity Scan 2010 Open Source Integrity Report for Android [6] done using static analysis testing (a type of attack) found 0.47 defects per 1,000 Source Lines of Code (SLOC) for a total 359 defects. Of these, 88 were considered "high risk" in the security domain. Also, an OS hole in Android with Angry Birds counterfeit software allowed researchers Jon Oberheide and Zach Lanier (<http://jon.oberheide.org/>) to gain access to embedded devices. Further, there are numerous reports [2] of cars and medical devices now being hacked. Additionally, soldiers and employees are carrying personal devices (smart phones) into the theatre and work environments that cannot be certified "risk free." With

security attacks such as GPS spoofing and cracking attacks, these devices introduce a risk brought in by the very users of the devices. And finally, there is the Stuxnet virus (<http://spectrum.ieee.org/podcast/telecom/security/how-stuxnet-is-rewriting-the-cyberterrorism-playbook>) and its decedents which demonstrate risks to factory control systems worldwide. These data points illustrate the risk of vulnerabilities to mobile and embedded devices. Teams need to be aware of traditional IT security approaches as well as those for mobile and embedded considerations.

Potential Solutions

For every security counter measure solution, hackers may find another problem or attack to implement. It is a constant game of cat and mouse. It is tempting to go back to pen and paper but even these were not secure! There are numerous general security safeguarding measures to consider, which are covered in this article.

If teams could create perfect software, they would not need to be concerned with security vulnerabilities because they could just "build it" secure. But since software development is a human creative process and humans make mistakes at all levels of development (concept and requirements, architecture and design, coding, and testing), teams can only seek to minimize security risks during these efforts.



**CIVILIAN TALENT IS MISSION-CRITICAL.
LET'S GET TO WORK.**

Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.

You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.

Discover more about NAVAIR. Go to www.navair.navy.mil.

Equal Opportunity Employer | U.S. Citizenship Required

**NAVAIR
CIVILIAN**

CHOICE IS YOURS.

Table 1: A Sampling of Secure Implementation Practices in Mobile and Embedded Development

Implementation Practice	Examples
Secure code generation	Teams should check for back doors, malware, and error prone code constructs
Defensive coding	Programmers initialize variables, bound and check tables, and logic redundancy [common mobile and embedded errors per 2]
COTS and open source code (to avoid counterfeit code)	Team uses wrappers to protect code, limits visibility of variables e.g., few if any global variables, conducts acceptance-security testing on mobile and embedded non developed code, focuses on “unique” interactions of one-of-kind “new” hardware with existing software, signatures, trusted vendors, anti-tamper, encryption, etc.
Static code analysis (SCA)	Testers use static code analysis tool to identify errors with tools containing specific checks for security issues and mobile and embedded error types
Developer testing	Programmers use object/unit test, coverage metrics, test first, Note: Many embedded systems require higher (than IT) levels of structural code coverage [15]
IEEE and ISO standards such as 1012, 12207, 15288 and 29119	Team follows industry standards respectively, and many mobile and embedded device maybe highly regulated with several compliance standards
Project coding standards	Project has locally defined coding standards or checklists that are followed and tailored for mobile and embedded coding practices
Analysis practices including software reliability engineering [16]	Team conducts threat analysis, risk analysis, failure mode effect analysis (FMEA) and failure mode effects and criticality analysis (FMECA), software reliability engineering, performance analysis, and so forth with a focus on unique aspects of hardware, software and operations.
Model or math-based engineering	Team uses models to generate code; use math principles in coding and logic, which is more common in critical embedded systems

Development Pattern Activity 1: Plan and Specify a Secure System [12]

Certainly building a more secure software system must start at the beginning with plans, concepts, and system specifications. The more security risks a software system has, the more these efforts can be justified. For example, consider:

- Methods to clearly specify requirements - see <http://sun.nyday.mit.edu/safer-world/index.html>
- Formal methods - see https://www.ece.cmu.edu/~koopman/des_s99/formal_methods/
- Model-based systems and software development - see Object Modeling Group at <http://utp.omg.org/>

Such approaches may be of use for some mobile and embedded systems, but more than likely, most systems with software efforts will use a waterfall lifecycle or derivative of it, or alternatively, they will use agile concepts [13]. However, any approach can introduce vulnerabilities. Additionally, most mobile and embedded systems are a mix of non-developed (off-the-shelf) and customized software components, both of which come with vulnerabilities. Organizations need to put forth a good offense and defense starting with systems engineering. Active Systems Engineering addresses functional and non-functional requirements, including security characteristics. Systems engineering should be supported by the right level of system analysis including active verification and validation with early attack testing, which can find some errors even before coding starts.

Finally, planning should address product control concepts such as anti-tamper features, software protection for downloaded files to ensure “correct” software is obtained e.g., internal signatures such as a checksum, encrypted files and data, and trusted supply chain management. These concepts can be used to reduce the likelihood of a project obtaining counterfeit software from outside parties (i.e., malware). Afterwards, plans would progress into design, architecture, coding, and testing where each stage would ensure that steps are taken so that malicious or vulnerable software is not introduced into the product.

Development Pattern Activity 2: Design and Architecture [14]

As the system is planned and specified, considerations regarding architecture and design need to be included. In architecture and design, the engineer needs to develop secure software with proven practices including:

- Selecting secure architecture structures;
- Careful development of data structures and databases;
- Trade studies and specification of non-developed items;
- Minimization of design coupling and maximized cohesion;
- Involvement of user considerations, where user includes the malicious user or hacker;
- Rigorous systems and software engineering practices including threat and risk assessment; and
- Design practices that consider security threats.

Weak architecture and design practices can impact the overall vulnerabilities that no amount of good coding and test practices can fully overcome.

Development Pattern Activity 3: Coding Practices [2]

With foundations of requirements and frameworks provided by architecture and design, implementation programming stands a better chance of minimizing errors and vulnerabilities. All projects need good coding practices in place to help minimize sloppy coding practices which can introduce risks. Samples of recommended mobile and embedded coding practices are listed in Table 1.

Development Pattern Activity 4: Verification and Validation Security Testing

A significant focus of this paper is on concepts that can be applied to mobile and embedded systems at any point in time to improve and address security vulnerabilities by applying software verification and validation testing concepts as defined in standards such as IEEE 1012 and ISO29119 [17,18]. Verification and validation security testing is applicable during develop-

ment, operation, and maintenance. For example, the security threats of a piece of mobile software will likely evolve as the devices are used in differing situations and for newer purposes in operational use. Consider a smart phone app designed for civilian use that is taken into the battlefield without proper security testing only to end up compromising, hurting or killing our troops or our "friendlylies." As part of full life cycle rigorous security practices, the concepts of risk and attack-based testing [18,2] prove particularly useful.

Risk-based Testing with Criticality Levels [18]

Fundamental to many development and testing efforts is the idea of risk management. Not every mobile or embedded software device poses the same risk, threat, or has the same vulnerabilities. For example, a small standalone mobile game on a smart phone is not the same as a mobile mapping app that a soldier may be using, but when on this person in a given situation, this smart phone and the app may be a security threat. Understanding the risk directs much of the development and testing efforts. IEEE 1012 defines integrity levels that can be used to determine the nature of verification and validation and/

or testing. ISO29119 uses risk-based testing to determine test plans, design, levels, and techniques. Basically, more risk in the security or quality areas would mean more testing should be done.

Currently, many mobile and some embedded software systems opt for "no security risk." This may mean no or minimal testing. However, the use of such software in different operational uses (i.e., networked) can mean more testing is needed. Further, the use and incorporation of non-developed (off-the-shelf) software may introduce the risk of counterfeit software parts. When risk analysis indicates security concerns, rigorous verification and validation with attack-based testing are often indicated.

Attack-based Testing [2, 19,20,21]

A test attack is a pattern to approach testing based on common modes of failure seen over and over. Attacking software and systems is an attempt to demonstrate that they (hardware, software, and operations) do not meet requirements or functional and non-functional objectives. Attacks target errors as well as provide other valuable information to stakeholders. Because

Table 2: Software Security Testing Attacks to Use in a Robust Risk-Based Approach (summarized and excerpted from [2])

Named Attack	Apply Against	Example Considerations
Penetration Attack	Account numbers/user ids	Use tools to gain access e.g., pkcrack
	Passwords	Check common passwords that may be "shortened" due to mobile device characteristics such as screen size, no keyboard, etc.
	Usage profiles	How is the device's usage profile or data being used in mobility?
	Location tags	Where is the device, are tags temporary as the device moves, and what is reported to an open network (cellular, Wi-Fi, etc.)?
Fuzz Testing Sub Attack	External inputs e.g., userids passwords	Use fuzzing tool to attack the mobile and embedded interfaces
Spoofing Attack	"Hijacked" Identity	Use spoofing tools in mobile and embedded "sand box" test environments (a sand box is separated from the full networked world so testing can be done "safely")
	User profile spoof	Lab environment setup is important. For example, a tester should consider using software based simulators when testing phones/device-ids since many apps key on this in the hardware and then the app "locks out" a particular device (device-id) when it is used in security testing more than a few times. This increases testing costs because a string of new devices must always be on hand and used to complete security testing.
	GPS spoof	Requires specialized equipment and labs. But for devices dependent on GPS, this may be a "high" risk factor
	"Social Engineering" spoof	Attack like the hackers, who are shifting their focus to mobile and embedded systems
Checking attack	"Hidden" files with unsecured data	Many mobile and embedded devices have a file structure allowing files to be hidden by programmers; files may not be easy to find unless a tester or hacker knows where to look
	Encryption (or lack thereof)	Is there restricted data perhaps hidden in mobile and embedded file systems which may be "temporary" and/or not encrypted properly?
	Good encryption patterns	Where did the algorithm(s) come from?
Breaking Software Security	Use classic IT/PC/web attacks many of which are applicable to mobile and embedded	See Whittaker's book [20] for 20 attacks that can be applied to mobile hybrid/web apps
Virus Attack	Off-the-shelf software	Test for counterfeit logic such as mobile and embedded viruses, malware, etc.
	Third party software	Many viruses are embedded in "fun" apps that users download particularly on "bring your own devices"
	Operating System	Can it be trusted?
	Bring your own device	Threat from unsecured users
	Battery life	Are batteries being depleted unexpectedly due to virus "usage?"
	Embedded multi-tier system	For example Stuxnet and its offspring

ABOUT THE AUTHOR



Jon Hagar is a senior systems-software engineer and testing consultant supporting software product integrity, verification, and validation with a specialization in mobile and embedded software systems. Jon is the lead editor/author on ISO 29119 Software Testing series of Standards, co-chair of the OMG UML Test Profile

Standard model based test standard, and contributor to IEEE 1012 verification and validation plans.

Phone: 303-903-5536

Email: embedded@ecentral.com

Website: <http://breakingembeddedsoftware.com>

REFERENCES

1. MSK McClure, Scambray, and Kurtz, "Hacking Exposed" McGraw Hill
2. JDH - J. Hagar "Software Test Attacks to Break Mobile and Embedded Devices", CRC press, 2013
3. Smart Phone Market Growth, ABI research report , 2013
4. C. Miller & C. Valasek, "Adventures in Automotive Networks and Control Units", web document <www.exploit-db.com/download_pdf/27404/>
5. . Security threats on embedded consumer devices, OMTech report, 2009
6. J. Weiss, "Threats impacting the nation" Testimony before the subcommittee on oversight, investigations, and management, committee on homeland security, House of Representatives, US Government accounting office, Washington, D.C.
7. Coverity Scan 2010 Open Source Integrity Report for Android, see <<http://www.coverity.com/company/press-releases/read/coverity-scan-2010-report-reveals-high-risk-software-flaws-in-android-html>>
8. McAfee, "McAfee Threat Report: 4th quarter 2012", McAfee Labs Pub, Santa Clara, CA, 2012
9. M. Arapinis, L. Mancini, E. Ritter, M. Ryan, N. Golde, K Redon, and R. Borgaonkar, "New Privacy Issues in mobile telephony: Fix and Verification in computer and communications security", ACM, vol/no: missing, pp 205-216, 2012
10. D. Bhasker, "4G LTE Security for Mobile Network Operators", Cyber Security and Information Systems , Vol 1 No 4, pp 20-29
11. K Venkatasubramanian, E Vasserman, O. Sokolsky, and I Lee, " Security and Interoperable Medical Device systems", IEEE security and privacy, Vol 10, No 5, 2012
12. M. Merkow, and L Raghavan, "Secure and resilient software", CRC press, 2012.
13. Agile software - <<http://agilemanifesto.org/>>
14. D Kleidermacher "Embedded Systems Security, Practical Methods for Safe and Secure Software and Systems Development" Newnes books, ISBN :9780123868879, 2012
15. R Mahmood, "A whitebox approach for automated security testing of android applications on the cloud", 7th international workshop on automation of software test , 2012
16. Taz T. Daughtery, "Security systems through software reliability engineering", Cyber Security and Information Systems, Vol 1, No 1, 2012
17. IEEE1012 Verification and Validation Plan, IEEE press, 2012
18. ISO/IEEE/IEC29119 Software Test Standard, IEEE/ISO press, 2013
19. W.J.W.Z. Chuanxiong Guo, "Smart phone attacks and defenses" Microsoft Research Pub
20. . J. Whittaker & H. Thompson "How to Break Software Security" Addison Wesley, 20049.
21. M.A. Mobarhan, M.A. Mobsrhan, and A. Shahbahrani, "Evaluation of security attacks on different mobile communication systems", Canadian Journal on Network and Information Security, Vol 3, No 1 Aug 2012
22. J Viega and H. Thompson, "The state of embedded device security", IEEE security and privacy, Vol 10, No 5, 2012

testers usually only run checks to verify that the system or software meets requirements, which is necessary, but not sufficient, test attacks should be practiced often. Pure requirements-based testing can miss large (egregious) errors and vulnerabilities that can be leveraged to allow access to a system.

Some may see test attacks as a negative. However, attacks can be viewed as a positive for security testing since these are the methods hackers are employing and if the efforts can stop any hacking, that could be considered as a positive as well as worthwhile. Using the information from an attack, developers can then improve the overall security of software or systems.

Mobile and embedded attacks were developed for errors determined from a historic industry taxonomy database [2]. The attack patterns use classic test and security evaluation techniques. A taxonomy is a classification of error patterns. IEEE has offered research over the years that many security testers and hackers form mental models of system failures and then, learn patterns to find these commonly occurring errors. Attack patterns build on mental models to aide security testers in a particular domain, herein for mobile and embedded devices.

This article offers examples of mobile and embedded security attacks by name although the details are out of scope of this piece (see [2] for specific actions and details). These attack patterns are based on researched industry taxonomy, which was created over a large number of publicly reported security errors and flaws. Taxonomies and attack patterns will never be comprehensive or complete since the nature of systems and software is evolving and not every project makes public the details of their vulnerabilities much less how they are found. Readers should use Table 2 as an introduction then, continue their own research into taxonomies and attack patterns, which may fit their local mobile and embedded contexts.

Table 2 is a beginning for mobile and embedded security testers. Many of these attacks relate to traditional security testing concepts, but the examples cite specific concerns that mobile and embedded testers should consider. The security test world is a fast moving and ever changing area. Newer threats and error taxonomy patterns are emerging constantly. Mobile and embedded security testing projects must constantly research, learn, and improve to stay current with what hackers are doing and to understand future vulnerabilities.

Summary

The concepts and attacks in this paper should be viewed as a starting point for projects wishing to improve their secure software development approaches, given the current poor practices [22]. Both doing and not doing these concepts can have impacts on the quality as well as on legal considerations. Mobile and embedded device use, features, and connections will continue to increase in the world and could mean that security threats will also increase. Security concerns impact all engineering domains e.g., system, hardware, software, and support areas such as testing. Mobile and embedded projects should be proactive during development using concepts such as attack-based testing to reduce many security vulnerabilities.

Upcoming Events

Visit <<http://www.crosstalkonline.org/events>> for an up-to-date list of events.

Better Software Conference East

Nov 8- Nov 13, 2015
Orlando, Florida
<http://bsceast.techwell.com>

30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)

November 9-13, 2015
Lincoln, Nebraska
<http://ase2015.unl.edu/#tab-main>

ACTION15: Actionable Analytics for SE

Nov 9, 2015 – Nov 13, 2015
Lincoln, Nebraska
<http://action15.github.io>

2015 IEEE 23rd International Conference on Network Protocols (ICNP)

Nov 10, 2015 – Nov 13, 2015
San Francisco, CA
<http://icnp15.cs.ucr.edu>

INFuture2015: e-Institutions – Openness, Accessibility, and Preservation

Nov 11, 2015 - Nov 13, 2015
Zagreb, Croatia
<http://infoz.ffzg.hr/INFuture>

2015 SC - International Conference for High Performance Computing, Networking, Storage and Analysis

Nov 15, 2015 – Nov 20, 2015
Austin, TX
http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=32761

Software Solutions Conference 2015

Hilton Crystal City, Arlington, VA
November 16-18, 2015
<http://www.sei.cmu.edu/ssc/2015/>

IEEE Real-Time Systems Symposium

San Antonio, Texas
December 1-4, 2015
<http://2015.rtss.org/>

ReConFig 2015

Mayan Riviera, Mexico
December 7-9, 2015
<http://www.reconfig.org>

Cyber Defense Initiative 2015

Washington, DC
Dec 12-19, 2015
<http://www.sans.org/event/cyber-defense-initiative-2015>

13th Annual IEEE Consumer Communications & Networking Conference

Las Vegas, NV
January 9-12, 2016
<http://ccnc2016.ieee-ccnc.org/>

12th Annual Open Forum for Large-Scale Network Defense Analytics

Daytona Beach, FL
January 11-14, 2016
<http://www.cert.org/flocon/>

International Conference on Verification, Model Checking, and Abstract Interpretation 2016

St. Petersburg, FL
January 17-19, 2016
<http://conf.researchr.org/home/VMCAI-2016>

ICCMS 2016: the 7th International conference on Computer Modeling and Simulation

Brisbane, Australia
Jan 18-19, 2016
<http://www.iccms.org/>

MODELSWARD 2016- The 4th International Conference on Model-Driven Engineering and Software

Rome, Italy
Feb 19-21, 2016
<http://www.modelsward.org/>

It's About Time!

I am an Air Force Brat – in fact, there's been only a total of 24 days of my life that I did not have a military ID card (I've had dependent, active duty, and retired). You have a different viewpoint on life if you were a military dependent and grew up overseas. You appreciate some things a whole lot more (I lived most of my childhood without television – I always felt it left scars). One thing I appreciate is time.

I grew up in Turkey - Istanbul, to be precise. Istanbul was a wonderful city to grow up in, full of history. Istanbul was the head of three empires – Eastern Roman, Byzantine, and Ottoman. The city had history everywhere you looked! Also, Istanbul is one of the few cities that span two continents (the Bosphorus separates the city into two parts – one in Europe, one in Asia). We lived there during the 1960s – and my dad (like his dad before him) was possibly a tiny bit OCD about “time.” When my dad was a very young man, he remembers his dad buying him his first wristwatch. My dad, in turn, bought me a wristwatch when I was 8. I will admit that I was forever breaking it – he seemed to have patiently re-bought me one every few years. He even bought me yet another new one when I discovered two things: 1) a strong magnet from an old speaker will stop a mechanical wristwatch; and 2) removing the magnet does not make it start again.

The problem (when living in Turkey) was that there was no easy way to accurately set the time. Today, of course, we are used to having cell phones (with precisely accurate times), plus numerous other ways to determine the right time (down to the millisecond, if needed): TV, the internet, radio, etc. We had none of these in Istanbul. But – we had a shortwave radio. It ran on 5 vacuum tubes – a Hallicrafters set he had bought back in the late 1950s when we lived in Scotland (I was born there). It picked up AM only - FM radio was not really popular until the 1960s. However, for those of you who understand how AM works – it picked up BOTH long and short-wave! Long wave is what we call “regular AM radio” – and shortwave? Well, that's where the magic came in. I could listen to stations from ALL OVER THE WORLD! The British Broadcasting Channel and BBC news. Australia. The Voice of America. I would wait around until it grew dark (shortwaves propagate or “bounce” at night) and twirl the dial and be transported all over the world.

And then my dad let me in on the secret of radio station WWV – Ft. Collins Colorado. Operated then by the National Bureau of Standards (now run by NIST – National Institute of Science and Technology). They broadcast (at 2.5, 5, 10 and 15 Mhz back then, as I recall) a time signal, accurate to the second. Dad and I used a tree, and ran a 7.5-meter antenna wire (which, he explained to me, was a quarter-wavelength signal for 10 Mhz. I found this relationship amazing, and learned how to convert Hz to wavelengths). We just had to remember to disconnect the antenna connection every time it clouded up (hate to have a lightning strike blow the radio!).

So, every Saturday night, it was our ritual. We'd turn on the radio, tune in WWV, and set all of the watches and clocks in the house. It was something that Dad and I did together.

Time is important, of course. If things occur out of order (too early or too late), “time” itself becomes wasted. I traveled to Baltimore this summer to attend a meeting – and I wanted to visit a few relatives along the route. I drove up and made the following observation: Note to self: Update the map on the GPS BEFORE you leave – because knowing the detours for construction coming into Baltimore from Washington DC would really be nice.

It's all about time.

This issue is about “Fusing IT and Real-Time Tactical.” I was trying to think of how much you have to understand about technology to just appreciate topic. Information technology used to be the domain of punch cards and grey-haired COBOL programmers who were one step away from retirement. Real-time development used to be the domain of a few weirdo's in the closed rooms who ignored modern compilers and hardware, and who hand-coded machine code that ran on (maybe) 8-bit processors. They had NOTHING in common. Now, both are critical technologies with complex interrelationships.

Things change, and I repeat, it's all about time.

My introduction to shortwave radio technology occurred almost 50 years ago. Back in 1997, I gave Dad a birthday gift of a modern “atomic” clock that adjusted itself every night. He marveled over it constantly. A few years ago, I bought myself an “atomic” wristwatch. I can't think of anything I do that requires accuracy of +/- 0.5 seconds, but you never know. Every time I look at my watch, I think back to the days of the shortwave radio in Istanbul, and smile a little.

My Dad passed away three years ago – and to the very end, he and I argued about which wristwatch was the best. I now teach college – and it's common to have a class of students where nobody actually wears a watch anymore. They all use their cellphone. I, in turn, would give almost anything to be back in Istanbul, wristwatch in hand, with my Dad, slowly tuning the radio to search for the voice saying, “.....tick....tick....at the tone, the time will be 20 hours, 50 minutes coordinated Universal Time.” I still have the Hallicrafters radio. It's a reminder of special times. I didn't realize how special those times were until they were long past.

It's all about time. You shouldn't waste it – whether it's IT, tactical, real-time or life.

David A. Cook
Professor of Computer Science
Stephen F. Austin State University
cookda@sfasu.edu

CROSSTALK / 517 SMXS MXDED

6022 Fir Ave.
BLDG 1238
Hill AFB, UT 84056-5820

PRSR STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737



CIVILIAN TALENT IS MISSION-CRITICAL. LET'S GET TO WORK.



Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.

You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.

Discover more about NAVAIR. Go to www.navair.navy.mil.

Equal Opportunity Employer | U.S. Citizenship Required



CHOICE IS YOURS.



NAV  AIR



CROSSTALK thanks the
above organizations for
providing their support.